НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК 004.38

«До захисту допущено»

Завідувач кафедри СПСКС

_____ В.П.Тарасенко_
(підпис) (ініціали, прізвище)

"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
Комп'ютерні системи та компоненти

на тему ВЕБ-ОРІЄНТОВАНА СИСТЕМА ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ СТУДЕНТСЬКОЇ ПОЛІКЛІНІКИ КПІ ім. ІГОРЯ СІКОРСЬКОГО

(АПАРАТНА ЧАСТИНА)

Виконав: студент ІІ курсу, групи КВ-71мп
(шифр групи)

Гриценко Микита Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н., доцент Клятченко Я.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (спеціалізація) – 123 Комп’ютерна інженерія

Комп’ютерні системи та компоненти

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ В. П. Тарасенко

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Гриценку Микиті Сергійовичу

1. Тема дисертації «Веб-орієнтована система для автоматизації роботи студентської поліклініки КПІ ім. Ігоря Сікорського (апаратна частина)», науковий керівник дисертації Клятченко Ярослав Михайлович, к.т.н., доцент, затверджені наказом по університету від «30» жовтня 2018 р. №4030-с

2. Термін подання студентом дисертації «__» грудня 2018 р.

3. Об’єкт дослідження є методи та системи автоматизації роботи медичних закладів.

4. Предмет дослідження є апаратна реалізація розробленої системи автоматизації роботи медичних закладів.

5. Перелік завдань, які потрібно розробити:
   - провести аналіз існуючих систем автоматизації медичних закладів;
   - дослідити комплектуючі для створення апаратної системи;
   - розробити клієнт-серверну архітектуру апаратних компонуючих;
   - розробити апартане забезпечення для демонстрації роботи системи;
   - проаналізувати результати роботи прототипу.

6. Перелік ілюстрованого матеріалу: презентація

6. Перелік публікацій:
   - Тези доповіді «Прикладна математика та комп’ютинг» ПМК-2018-2
   - Тези доповіді на V-й Міжнародній науково-технічній Internet-конференції НУХТ, АКС 2018

7. Дата видачі завдання «04» жовтня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	25.10.2017	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	17.03.2018	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	25.04.2018	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	24.05.2018	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	16.06.2018	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	17.07.2018	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка матеріалів доповіді на конференції ПМК-2018	21.10.2018	
8.	Оформлення текстової і графічної частини магістерської дисертації	15.11.2018	

Студент М. С. Гриценко

Науковий керівник дисертації Я. М. Клятченко

РЕФЕРАТ

Актуальність теми. Кожен студент декілька разів за період навчання відвідує студентську поліклініку при навчальному закладі. Зазвичай це викликає ряд певних незручностей. Потрібно заздалегідь дізнатися який лікар обслуговує факультет на якому навчається студент та дізнатися графік роботи цього лікаря. Також потрібно прийти до поліклініки за декілька годин до запланованого візиту щоб зайняти чергу на прийом. Графік роботи лікарів постійно змінюється, а інформація про подібні зміни не публікується на інтернет-ресурсах.

Для вирішення цих проблем потрібно впровадити веб-орієнтовану електронну медичну систему для автоматизації роботи студентської поліклініки КПІ ім. Ігоря Сікорського. Ця система повинна включати в себе зручний веб-інтерфейс з усією актуальною інформацією про поліклініку та програмно-апаратний комплекс, що дозволяє формалізувати та оптимізувати управління потоком відвідувачів — електронну чергу.

Об’єктом дослідження є методи та системи автоматизації роботи медичних закладів.

Предметом дослідження є апаратна реалізація розробленої системи автоматизації роботи медичних закладів.

Мета дослідження: оптимізація управління потоком відвідувачів та зменшення їх часу відвідування установи.

Наукова новизна полягає в дослідженні та проведенні навантажувальних тестувань мінікомп’ютерів та мікроконтролерів, створенні їх порівняльних характеристик та розробки апаратної складової медичної інформаційної системи на основі отриманих результатів

Практична цінність полягає у впровадженні отриманих в роботі результатів на практиці в студентській поліклініці КПІ ім. Ігоря Сікорського

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на XI науковій конференції молодих вчених «Прикладна математика та комп'ютинг» ПМК-2018-2 (Київ, 14-16 листопада 2018 р.) та на V Міжнародній науково-технічній Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами» в НУХТ на базі факультету АКС.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, п'яти розділів, висновків та додатків.

У вступі надано загальну характеристику роботи, описано проблематику та сформульовано мету дослідження,

У першому розділі зроблено короткий огляд існуючих методів вирішення задач автоматизації медичних установ, представлено загальне поняття таких систем та їх короткий аналіз.

У другому розділі проведено дослідження та тестування мікроконтролерів та міні комп'ютерів; обрано апаратні рішення за результатами тестування.

У третьому розділі проведено дослідження та порівняння мережевих протоколів передачі даних та методи захисту інформації.

У четвертому розділі розроблено та описано апаратні рішення систем автоматизації медичних закладів; описано структуру апаратної частини систем автоматизації медичних заклади; описано алгоритм роботи системи.

У п'ятому розділі проаналізовано розроблену систему та описані інтерфейси для роботи з нею.

У висновках проаналізовано отримані результати роботи.

Магістерська дисертація виконана на 90 аркушах, містить _?_ додатків та список використаних літературних джерел з 20 найменувань. У роботі наведено 40 рисунків та 38 таблиць.

Ключові слова: автоматизація, апаратна система, електронна черга, Raspberry, ESP, Arduino, мікроконтролер.

ABSTRACT

Theme urgency. Each student attends a student clinic several times during the period of study. This usually causes a number of inconveniences. Student needs to know in advance what kind of doctor serves the faculty and to find out the schedule of this doctor. Student must come to the clinic a few hours before the scheduled visit to take a queue at the reception. The schedule of doctors' work is constantly changing, and information about such changes is not published on Internet resources.

To solve these problems, web-based electronic medical system for automation the reception of the student should be implemented. This system should include a convenient web interface with all relevant information about the clinic, software and hardware complex, which allows to formalize and optimize the visitors flow management - an electronic queue management system.

Object of research is the methods and systems of medical institutions automation.

Subject of research is the hardware implementation of the developed system for medical institutions automation.

Research objective: optimize the visitors flow and reduce their time of visiting the clinic.

Scientific novelty consists in researching and conducting loading tests of minicomputers and microcontrollers, creating their comparative characteristics and developing the hardware component of the medical information system on the basis of the obtained results.

Practical value is the introduction of the results obtained in the work in practice at the student clinic of Igor Sikorsky Kyiv Polytechnic Institute.

Approbation. The basic points and outcomes of the research have been presented and discussed at the 11th scientific conference for students and postgraduates «Applied mathematics and computing» PMK-2018-2 (Kyiv, November 14-16, 2018) as well as at the 5th International Conference at ACS NUFT (November 24-23, 2018, Kyiv, Ukraine).

Structure and content of the thesis. The master thesis consists of the introduction, five chapters, conclusions and appendixes.

The introduction presents the general description of the research.

In the first chapter a brief overview of existing methods for solving the problems of automation of medical institutions was made, it presents the general concept of such systems and their brief analysis.

In the second chapter research and testing of microcontrollers and mini computers were conducted; hardware solutions are selected based on testing results.

In the third chapter research and comparison of network protocols and data protection methods were conducted.

In the fourth chapter hardware solutions for automation systems of medical institutions were developed and described; structure of the hardware part and the algorithm of the automation system of a medical institution were described.

In the fifth chapter developed system was analyzed and interfaces to work with it was described.

In the conclusions general conclusions and results of analysis of model work are presented.

The thesis is presented in 90 pages, it contains _ appendixes and 20 references to the used information sources. 40 figures and 38 tables are given in the thesis.

Key words: automation, hardware system, electronic queue management system, Raspberry, ESP, Arduino, microcontroller.

РЕФЕРАТ

Актуальность темы. Каждый студент несколько раз за период обучения посещает студенческую поликлинику при учебном заведении. Обычно это вызывает ряд определенных неудобств. Нужно заранее узнать какой врач обслуживает факультет на котором учится студент и узнать график работы этого врача. Также нужно прийти в поликлинику за несколько часов до запланированного визита чтобы занять очередь на прием. График работы врачей постоянно меняется, а информация о подобных изменениях не публикуется на интернет-ресурсах.

Для решения этих проблем нужно внедрить веб-ориентированную электронную медицинскую систему для автоматизации работы студенческой поликлиники КПИ им. Игоря Сикорского. Эта система должна включать в себя удобный веб-интерфейс со всей актуальной информацией о поликлинику и программно-аппаратный комплекс, позволяющий формализовать и оптимизировать управление потоком посетителей - электронную очередь.

Объектом исследования являются методы и системы автоматизации работы медицинских учреждений.

Предметом исследования является аппаратная реализация разработанной системы автоматизации работы медицинских учреждений.

Цель исследования: оптимизация управления потоком посетителей и уменьшения их времени посещения учреждения.

Научная новизна заключается в исследовании и проведении нагрузочных тестирований миникомпьютеров и микроконтроллеров, создании их сравнительных характеристик и разработке аппаратной составляющей медицинской информационной системы на основе полученных результатов

Практическая ценность заключается во внедрении полученных в работе результатов на практике в студенческой поликлинике КПИ им. Игоря Сикорского

Апробация работы. Основные положения и результаты работы были представлены и обсуждались на XI научной конференции молодых ученых «Прикладная математика и компьютинг» ПМК-2018-2 (Киев, 14-16 ноября 2018) и на V Международной научно-технической Internet-конференции « современные методы, информационное, программное и техническое обеспечение систем управления организационно-техническими и технологическими комплексами »в НУХТ на базе факультета АКС.

Структура и объем работы. Магистерская диссертация состоит из введения, пяти глав, заключения и приложений.

Во введении дана общая характеристика работы, описано проблематику и сформулирована цель исследования,

В первой главе сделано краткий обзор существующих методов решения задач автоматизации медицинских учреждений, представлены общее понятие таких систем и их краткий анализ.

Во втором разделе проведено исследование и тестирование микроконтроллеров и мини компьютеров; избраны аппаратные решения по результатам тестирования.

В третьем разделе проведено исследование и сравнение сетевых протоколов передачи данных и методы защиты информации.

В четвертом разделе разработаны и описаны аппаратные решения систем автоматизации медицинских учреждений; описана структура аппаратной части систем автоматизации медицинских учреждений; описан алгоритм работы системы.

В пятом разделе проанализирована разработанная система и описаны интерфейсы для работы с ней

В выводах проанализированы полученные результаты работы.

Магистерская диссертация выполнена на 90 листах, содержит _? _ приложений и список использованных литературных источников из 20 наименований. В работе приведены 40 рисунков и 38 таблиц.

Ключевые слова: автоматизация, аппаратная система, электронная очередь, Raspberry, ESP, Arduino, микроконтроллер.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

IoT	Internet of Things, Інтернет речей
Framework	готовий до використання комплекс програмних рішень
ПЗ	програмне забезпечення
БД	база даних
Скетч	блок коду для Arduino
IT	блок коду для Arduino
ОС	операційна система
SNMP	Simple Network Management Protocol
MQTT	Message Queue Telemetry Transport
REST	Representational State Transfer
SSH	Secure Shell
IP	Internet Protocol

ВСТУП

Кожен студент хоча б декілька разів за період навчання відвідує студентську поліклініку при навчальному закладі. Зазвичай це викликає ряд певних незручностей. Потрібно заздалегідь дізнатися який лікар обслуговує факультет на якому навчається студент та дізнатися графік роботи цього лікаря. Також потрібно прийти до поліклініки за декілька годин до запланованого візиту щоб зайняти чергу на прийом. Графік роботи лікарів постійно змінюється, а інформація про подібні зміни не публікується на інтернет-ресурсах.

Для вирішення цих проблем потрібно впровадити веб-орієнтовану електронну медичну систему для автоматизації роботи студентської поліклініки КПІ ім. Ігоря Сікорського. Ця система повинна включати в себе зручний веб-інтерфейс з усією актуальною інформацією про поліклініку та програмно-апаратний комплекс, що дозволяє формалізувати та оптимізувати управління потоком відвідувачів — електронну чергу.

Метою магістерської дисертації є проведення систематизації, аналізу, порівняння та тестування існуючих на ринку популярних мікроконтролерів та міні комп’ютерів та реалізація веб-орієнтованої електронної медичної системи. Система буде реалізована на оптимальних компонентах, що будуть визначені в результаті дослідження. При розробці системи буде враховано досвід інших медичних установ та потреби самих відвідувачів студентської поліклініки. Це дозволить позбавитися від живих черг під кабінетами лікарів і дати можливість відвідувачам спланувати час відвідування установи заздалегідь.

1. АНАЛІЗ СИСТЕМИ АВТОМАТИЗАЦІЇ МЕДИЧНИХ ЗАКЛАДІВ

1.1 Комплексна система автоматизації діяльності медичного закладу

Однією комплексних систем автоматизації діяльності медичного закладу є система збору інформації про пацієнта ПЗ МІС.

Одним із напрямів побудови медичних інформаційних систем є медичні інформаційні системи (МІС) . Ними оснащуються різні медичні заклади та їх підрозділи. До складу таких систем входять автоматизована реєстратура, формалізовані амбулаторні карти прикріпленого контингенту, облік і аналіз відвідуваності, захворюваності, профілактичних оглядів, диспансеризації, тимчасової непрацездатності, щеплень, флюорографічних досліджень, планування й облік роботи лікарів, формування всієї звітно-статистичної документації про діяльність поліклініки, стаціонару і їх підрозділів.

МІС забезпечують управління персоналом, фінансами, матеріально-технічними ресурсами, зокрема медикаментами, медичними інструментами та апаратурою. Якщо розглядати єдиний медичний простір з позиції пацієнта, то його основу становить електронна історія хвороби як важлива складова МІС. Лікар зможе одержувати оперативний доступ до необхідної медичної інформації за наявності електронної історії хвороби або за допомогою індивідуальної електронної медичної картки пацієнта незалежно від того, де перебуває пацієнт, в який медичний заклад він звернувся або був госпіталізований (державний або приватний).[1]

Широке впровадження МІС у систему схорони здоров'я України має велике значення для ефективного управління лікувально-профілактичним закладом, підвищення рівня якості й об'єктивності діагностики.

Система компонентів МІС в медичній установі (надалі - Установа) та її технічна підтримка, реалізована комплекс заходів щодо несанкціонованого та неконтрольованого ознайомлення, модифікації, знищення, копіювання, по-ширення даних.

Апаратно-технологічна частина ПЗ - забезпечена конфігурацією локальної мережі Установи та технічними засобами, на яких функціонує система МІС, а саме:

- база даних системи МІС розміщена на комп'ютері, який виконує роль центрального накопичувача інформації (сервер) в відповідній директорії, яка закодована паролем. Користувач комп'ютера на може «ввійти» в цю директорію та провести копіювання, корекцію або вилучення даних загальносистемними засобами

- всі робочі станції підключені до центрального комп'ютера (сервера) через локально-обчислювальну мережу (надалі - ЛОМ), яка розміщена виключно в приміщеннях та на території Установи. Можливість несанкціонованого підключення («врізки») до ЛОМ виключена

технічне відключення на комп'ютерах, які підключені до системи МІС, USB-портів. Цим заходом блокується можливість копіювання будь-якої інформації з ЛОМ Установи та можливість «занесення» вірусів в ЛОМ та систему МІС

- технічно контур ЛОМ Установи та контур комп'ютерів, що мають доступ до мережі Інтернет, розірвані. Таким чином виключена можливість проникнення в систему Установи через мережу Інтернет та занесення вірусів

- за регламентом, щоночі автоматично виконується процедура резервного копіювання даних системи МІС на зовнішній носій, визначений Адміністрацією. Резервна копія знаходиться в закодованому вигляді

- контроль загального стану БД та виконання її резервного копіювання, перевірка функціонування системи МІС, корекція можливих технічних помилок, якщо такі будуть виявлені, проводиться технічним фахівцем (представником Розробника) у відповідності до узгодженого з Адміністрацією регламенту

- при виконанні робіт з програмно-технічного супроводу технічний персонал не має доступу до персональних даних пацієнтів Установи

Програмно-інформаційна частина ПЗ - забезпечена програмними можливостями компонентів МІС та структурою БД, а саме:

- доступ до даних системи мають виключно ідентифіковані та

аутентифіковані користувачі. Кожний користувач має пароль доступу, який визначає його функції в системі МІС та доступ до даних. База паролів автоматично кодується

в системі МІС реалізований принцип, по якому користувачеві, який не має права доступу до будь-яких даних або функціональних режимів, система МІС навіть не відображає їх наявність

відповідно до повноважень та заданих прав в системі МІС, кожний користувач може переглядати дані, модернізувати, видаляти (видалення даних з системи реалізовано виключно за «логічним» принципом) або формувати узагальнені звіти та аналітичні таблиці, експортувати звіти в зовнішні програмні засоби

- в системі МІС не передбачені такі функції, як знищення, копіювання, поширення даних

- кожна дія користувача з даними автоматично реєструється в системі МІС з фіксацією імені користувача, дати та часу виконання операції. При цьому кожна карточка даних містить інформацію щодо її створення та останньої модифікації. Знищити цю службову інформацію не можливо

- в БД системи МІС кодується сама важлива інформація, а саме: доступ до директорії серверу, де знаходиться система МІС та її БД, паролі користувачів, дані пацієнта (прізвище, дата народження, ІНН, номер посвідчення особи, місце реєстрації, місце проживання, місце роботи, номер амбулаторної карти, діагнози) посилка (файл) обміну інформацією між дистанційно рознесеними вузлами МІС, резервна копія БД на зовнішньому носію

- за рішенням Адміністрації, в системі МІС реєструється інформація про надання письмової Згоди пацієнтом на обробку його персональних даних та Повідомлення пацієнту про занесення його даних в систему МІС

- система МІС містить функцію, яка дозволяє за заявою пацієнта, надати повну інформацію про його дані, що містяться в БД (амбулаторна карта хворого ф-025/о) у наступних формах - друкована, формат MS’Word, електронний медичний паспорт. При цьому в системі МІС проводиться реєстрація вказаної

процедури (користувач, дата, час, пацієнт, форма представлення) та формування відповідного документу, який підписує особа, що отримала вказану інформацію

- цілісність БД автоматично перевіряється системою МІС в процесі її запуску. У разі виникнення порушень, користувачеві видається відповідне повідомлення, при цьому система МІС припиняє свою роботу

- система МІС безперервно «контролює» системний час комп'ютерного обладнання, та, у разі його невідповідності, видається відповідне повідомлення, при цьому система МІС припиняє свою роботу

Широкий набір функціонально орієнтованих прикладних програм МІС дозволяє створювати різноманітні інформаційно-обчислювальні мережі, орієнтовані на рішення всього спектра задач організації управління лікувальним і лікувально-профілактичним процесом у медичному закладі.

У закладі охорони здоров'я використовується комп'ютерна техніка для опрацювання фінансової документації. При цьому скорочується термін виконання фінансових операцій і зменшується число помилок.

Електронна історія хвороби забезпечує оперативний облік витрат, пов'язаних із діагностичними та лікувальними процедурами, використанням медикаментів і матеріалів, оплатою послуг медичного персоналу тощо, що має ключове значення для страхової медицини.

Медична Інформаційна Система (МІС) - це інструмент для визначення і планування всіх ресурсів медичного закладу, які необхідні для ведення лікувально-діагностичної, адміністративно-господарської, фінансової, сервісної діяльності та обліку в процесі надання медичних послуг.

Основними перевагами запровадження МІС в медичному закладі є:

Єдина база даних, яка дає можливість вести оперативний обік всіх ресурсів (матеріальних, людських, фінансових). Результатом є планування, аналіз ефективності та оптимізація використання наявних ресурсів.

Збільшення пропускної спроможності медичного закладу за незмінних ресурсів (за рахунок оптимізації процесів введення, пошуку, зведення та аналізу даних; швидшої взаємодії між підрозділами; планування завантаженості лікарів,

кабінетів, обладнання).

Підвищення якості обслуговування пацієнтів (зменшується час очікування за рахунок планування; збільшується ефективний час перебування пацієнта в госпіталі) і, як наслідок, зростання задоволеності пацієнтів.

Використання електронних медичних протоколів, можливість підрахунку ефективності лікування, зменшення вірогідності медичних помилок підвищує медичну якість послуг.

Можливість швидкого формування будь-яких звітів для прийняття обґрунтованих управлінських рішень (кількість наданих послуг, завантаженість лікарів та кабінетів, статистика пацієнтів за діагнозами, віком, статтю, і т.д.).

Ефективне управління складськими запасами медикаментів та витратних матеріалів за рахунок автоматизації процесу замовлення-списання та підтримки оптимального залишку.

Підвищення рівня безпеки та конфіденційності інформації завдяки запровадженню політики прав доступу до різних даних та для входу в систему, за паролем або за відбитком пальця.

Використання МІС для оперативного управління медичним закладом

МІС дає можливість керівнику бути в курсі поточних справ клініки. За її допомогою він має можливість:

Знати, наскільки злагоджено працює клініка в цілому, та її окремі підрозділи, бути в курсі всіх проблем. За допомогою програмного забезпечення дізнатися про різницю між назначеним і фактичним часом прийому. У разі значних відхилень графіка прийому пацієнтів з'ясувати причину зсуву та прийняти оперативні заходи щодо їх усунення. За умов паперового документообігу малоймовірно, що керівник отримає інформацію оперативно та зможе виправити ситуацію одразу і впровадити зміни для уникнення повторення ситуації в майбутньому.

Контролювати поточну (в реальному часі) та заплановану завантаженість лікарів (або будь-яких інших ресурсів). Наприклад, керівник, відкривши розклад або звіт про завантаженість лікарів, може легко порівняти щільність запису

пацієнтів до гінеколога 1 і гінеколога 2. У випадку, коли з невідомих причин за тиждень гінеколог 1 прийняв пацієнтів удвічі більше ніж гінеколог 2, а значної різниці не мало бути, напевно, варто з'ясувати, чому так відбувається.

Бути в курсі фінансових питань. Отримати оперативну інформацію про надходження в касу, зокрема, за різними послугами, та знати стан дебіторської заборгованості, отримати звіт про стан розрахунків зі страховими компаніями, фінансові звіти за будь-який період часу та прогнози щодо надходжень на майбутній період.

Контролювати кількість залишків медикаментів та витратних матеріалів, вчасно приймати рішення про нові замовлення відповідно до наявних грошових коштів.

Усі ці операції виконуються за лічені секунди, допомагають керівнику тримати «руку на пульсі» та вчасно виправляти небажані ситуації.

1.2 Аналіз електронних систем управління чергою

Системи управління чергою допомагають уникнути скупчення людей в місцях прийому відвідувачів і організувати «цивілізований» порядок обслуговування клієнтів. В основному застосовується для розподілу, оптимізації та обліку клієнтів в черзі і виклику їх до кас за допомогою звукового сигналу і візуального відображення індивідуального номера черги клієнта. Найбільш типові застосування подібних систем: каси з продажу ж / д і авіаквитків, сервіс центри з надання послуг на вокзалах, каси прийому платежів, державні установи, офіси великих фірм, банки, пункти реєстрації автотранспорту (МРЕО) та ін. Системи управління чергою виготовляються на базі електронних табло.

Електронна черга - це програмно-апаратний комплекс, який застосовується в багатьох областях для автоматизації процесу запису і організації черги на послуги, що надаються. Вона дозволяє значно зменшити час очікування для клієнтів, а також зменшує навантаження на персонал. У поточний момент електронна черга стає звичним явищем і зустрічається в багатьох місцях, таких як банки, відділення пошти, офіси великих компаній, державні установи, медичні

установи та ін.

До складу систем можуть входити різні компоненти: принтери для друку порядкового номера клієнта, пульти управління для операторів і адміністратора, групові інформаційні табло, табло оператора, інтерфейс з комп'ютером та ін.. Додатково можливе збереження інформації про роботу системи з функціями контролю і підрахунку статистики, що в багатьох випадках дозволяє мати повну інформацію по завантаженості пункту обслуговування клієнтів, роботі окремих операторів і ін.[2]

Над кожним робочим місцем оператора встановлено світлове табло, що відображає поточний номер черги клієнта, що обслуговується в даний момент. Звуковий сигнал супроводжує виклик нового клієнта.

Система, дозволяє не тільки ефективно керувати чергою, а й одночасно піклується про те, щоб кожен пацієнт отримав індивідуальне обслуговування в комфортній обстановці, коли ніхто не завадить конфіденційної роботі медичного працівника з клієнтом. У будь-який момент співробітники можуть отримати практично будь-яку статистичну інформацію про обслуговування клієнтів і роботі лікарів. Використання системи управління чергою дозволяє прискорити час обслуговування відвідувачів більш ніж удвічі

Чекати не любить ніхто - це завжди негативно впливає на якість послуги. Проте, це явище не таке вже й рідкісне в повсякденному житті. Фактично в діяльності будь-якої організації де-небудь та виникають черги.

А адже саме емоції, які виносять клієнти з черг, - головне, що визначає їхнє ставлення до рівня отриманого сервісу в цілому і перспективи повторного обслуговування в цьому місці.

Лімітована продуктивність завжди є слабким місцем для сфери послуг, оскільки послуги не можуть вироблятися заздалегідь і зберігатися до моменту, коли будуть затребувані. Сучасна клієнт орієнтована організація повинна намагатися розробити стратегію, яка привносить порядок, передбачуваність і справедливість в чергу.

Поряд з вирішенням завдань управління потоками клієнтів є можливість

формувати за допомогою цієї системи різні статистичні бази даних.

Застосування системи електронної черги в операційно-касовому залі забезпечує:

- впровадження сучасної технології обслуговування клієнтів, розподіл і оптимізацію потоків клієнтів;
- середній час очікування на кожен тип обслуговування за тривалістю робочого дня, тижня (у вигляді графіків і діаграм з різних відрізках часу);
- час обслуговування одного клієнта окремим оператором для визначення відносної ефективності роботи кожного операціоніста (для оцінки продуктивності праці кожного касира і планування роботи);
- статистика по типам обслуговування, що дозволяє оперативно визначати завантаження оператора і розподіляти їх по типу обслуговування.
- отримання оперативної інформації в реальному масштабі часу про поточну
- роботу кожного касира, кількості працюючих кас, кількості обслужених клієнтів, кількості клієнтів, які чекають в черзі і ін.

Існують різні модифікації системи від простої, що забезпечує тільки просування черги до системи з розширеними сервісними функціями.

Перебуваючи в «зоні очікування» відвідувачі можуть підготувати необхідні документи або просто переглянути наявні тут довідники і почитати газети. І все це за відсутності «живої» черги з усіма її негативними моментами. Звуковий сигнал, що привертає увагу при зміні інформації на табло, не дасть можливості клієнтам пропустити свою чергу. Є навіть можливість друкувати на талонах орієнтовний час обслуговування або середній час очікування. Завдяки цьому клієнт може при бажанні відлучитися із залу очікування у своїх справах і повернутися до моменту обслуговування. Це створює додаткову зручність при великому потоці відвідувачів.

Ще однією перевагою системи є наявність програми статистичного обліку. Вона фіксує роботу кожного оператора, дозволяє аналізувати зібрані дані і відповідним чином планувати і змінювати роботу відділу в залежності від дня

тижня, сезону і потреб підприємства. Ця програма збирає дані про кількість клієнтів, обслугованих кожним працівником і відділом у цілому в певний момент часу - годину, день, тиждень, місяць і т.д., а також час обслуговування і очікування.

У практиці розробки та реалізації систем електронного управління чергою існує великий діапазон їх функціональних і інженерних рішень.

Необхідність цивілізованого підходу до вирішення проблем, пов'язаних з обслуговуванням клієнтів, тільки підтверджує, що за такими системами майбутнє!

Електронна черга має дві головні переваги:

1. Комфорт клієнта: можна отримати інформацію про потрібну послузі, роздрукувати талончик із зазначенням часу виклику по електронній черзі, записатися на прийом, а також дати оцінку якості роботи співробітників компанії.

2. Комфорт персоналу: система управління чергою дає чітку картину розподілу навантаження на співробітників і дозволяє побудувати графік роботи без стресових навантажень.

1.3 Аналіз існуючих рішень

Незадовільною є ситуація з інформуванням міських управлінь охорони здоров’я, санепідемстанцій та інших установ про епідеміологічну ситуацію чи поточний стан захворюваності, та наявність вільних ліжок в лікарнях тощо. Через відсутність сучасної техніки, програмного забезпечення та засобів зв’язку така інформація є неповною і запізнілою, що не дає можливості оперативно та адекватно попереджати загрози, а також реагувати на проблеми, які виникають у роботі медичних закладів. Більшість медичних інформаційних систем, які функціонують у лікувальних закладах в даний час, є морально і фізично застарілими. Переважно вони розроблені ще 10-15 років тому, їх ніхто вже давно не підтримує і не удосконалює. Ці системи дозволяють автоматизувати тільки підготовку звітних форм. На сьогодні на ринку медичних інформаційних системи присутні 10-15 розробників. За кількістю впроваджень слід відзначити:

«Медсистеми», CIET, «Укрмедсоф», TherDep. До українського ринку проявляють інтерес також польські (ABG), російські(«Медиалог») та турецькі розробники медичних інформаційних систем. Проте вартість впровадження цих систем є значно вищою, ніж у аналогічних українських систем. Більшість систем побудовано на основі клієнт-серверної архітектури, яка забезпечує обмежену кількість функцій – переважно підготовку статистичних звітів та стандартних форм МОЗ. У цих системах ведеться електронна історія хвороби, внесення даних до яких здійснюється шляхом набору тексту або вибору фраз з довідників. Такий підхід не дає можливості в подальшому здійснювати поглиблений аналіз. Недоліком цих систем є необхідність звертатися до розробників для внесення змін у вхідні й вихідні форми. [3]

Приємно відзначити появу на ринку вітчизняних розробників систем, які підтримують 3-рівневу архітектуру. Це «Доктор Елекс» та «ЕмсіМед». Ці системи орієнтовані не тільки на державні, але й на орієнтовані не тільки на державні, але й на приватні медичні заклади. Вони забезпечують інтеграцію електронної карти пацієнта з різноманітним діагностичним обладнанням, а також забезпечують отримання даних безпосередньо з лабораторних аналізаторів. Внесення в електронну історію хвороби медичних даних здійснюється на основі розроблених лікарями-експертами протоколів. Це відкриває широкі можливості для подальшого всестороннього аналізу всіх даних. В цих системах є конструктор вхідних і вихідних звітних форм; вони забезпечують можливість обміну шаблонами документів.

На особливу увагу заслуговує медична інформаційна система «Доктор Елекс». Вона розроблена з врахуванням сучасних стандартів та принципів взаємосумісності медичних інформаційних систем. В основі системи лежить ідея побудови лікарських оглядів на базі деревовидних шаблонів оглядів.

Система забезпечує всі інформаційні потреби лікувально-реабілітаційного та діагностичного процесів, науково-дослідної та навчально-методичної роботи. Робота над створенням інформаційної системи в ТзОВ «Елекс» розпочалася ще в1990 році. Першою розробкою компанії у медичній галузі була система

«Авалон», впроваджена у ряді медичних закладів України. Подальший досвід компанія «Елекс» отримала при розробці онкологічної системи для університету міста Тампа, Каліфорнія, США та великої системи для збору статистики з використанням стандарту HL7 для американського ринку. Підсумком усіх інновацій стала система «Доктор Елекс», розроблена на найновіших технологіях із урахуванням досвіду і знань, отриманих фахівцями компанії під час роботи над попередніми системами. МІС дає можливість вводити в оптимальній формі, зберігати та аналізувати не тільки основні дані пацієнта, зазвичай використовувані у реєстратурі, а й усю медичну документацію, таку як скарги, анамнез життя і захворювання, дані об'єктивного обстеження, функціональної та лабораторної діагностики, антропометрії, а також дані про лікарські призначення та їх виконання впродовж перебування у лікувальній установі. Основним компонентом зберігання даних пацієнтів в інформаційній системі є електронна медична карта, в якій накопичується вся інформація: дані лікарських оглядів, антропометричні виміри, дані відео контролю, щоденники динамічного спостереження стану пацієнта, виписки та результати обстежень інших клінік, мультимедійні дані (рентгенограми, проби письма, фото) та інші важливі дані про пацієнтів. Основна медична інформація, така як дані лікарського огляду та результати лікування, вводиться в електронну карту згідно спеціально розробленої уніфікованої медичної термінології, яка організована у деревовидні шаблони огляду — ієрархічні структури, що складаються із примітивів, які формують логіку лікарського обстеження. Система пройшла незалежне тестування і рекомендується МОЗ до впровадження в медичних закладах. Впровадження інформаційних технологій в медицині заслуговує на безпосередню увагу керівників галузі і зацікавлених відомств. Одним з пріоритетних напрямів розвитку системи охорони здоров'я є створення єдиного медичного інформаційного простору, який забезпечить прийняття ефективних управлінських рішень на всіх рівнях. Це дасть можливість налагодити ефективний облік діяльності медичним закладам організації здійснювати на сучасному рівні менеджмент, своєчасно отримувати інформацію про передові досягнення в галузі

медичної науки, використовувати всю медичну інформацію про пацієнта (за весь період його життя), накопичену зі всіх рівнів надання медичної допомоги для досягнення кращого лікувального ефекту.

2. ХАРАКТЕРИСТИКА АПАРАТНИХ РІШЕНЬ ТА ІНСТРУМЕНТАРІЮ

Поява перших мікроконтролерів ознаменувала початок нової ери в розвитку мікропроцесорної техніки. Наявність в одному корпусі більшості системних пристроїв зробило мікроконтролер подібним до звичайного комп'ютера. В вітчизняній літературі вони навіть називалися одно кристальними мікроЕОМ. Відповідно і бажання використовувати мікроконтролери як звичайні комп'ютери постало практично з їх появою. Але бажання це стримувалося багатьма факторами. Наприклад, щоб зібрати пристрій на мікроконтролері, необхідно знати основи схемотехніки, пристрій і роботу конкретного процесора, вміти програмувати на асемблері і виготовляти електронну техніку. Потребуються також програматори, відлагоджувачі і інші допоміжні пристрої.

Мікроконтролер - це невеликий комп'ютер на єдиній інтегральній схемі, що містить ядро процесора, пам'ять та програмовані периферійні пристрої введення / виведення. Це високо інтегрований мікросхема, що містить всі компоненти, що складаються з контролера. Мікроконтролери використовуються в автоматично контрольованих продуктах та пристроях, таких як системи керування автомобільним двигуном, дистанційне керування, офісні машини, медичні пристрої, прилади та інші вбудовані системи.

2.1 Мікроконтролери Arduino

Фактично ера доступних та простих мікроконтролерів розпочалася з появою рішень від Arduino. Вдала реалізаціє разом з нескладним використанням та численними інструкціями по роботі з ними дозволили кожному бажаючому долучитися до роботи з мікроконтролерами. Згодом з’явилися плати розширення, які дозволили підключати їх до локальних мереж чи інтернету, що суттєво спростило реалізацію концепції IoT.

Arduino - це платформа електронних компонентів із відкритим кодом, що базується на простій у використанні апаратній та програмній продуктах.

Апаратна платформа Arduino вже має налаштування живлення та скидання схеми, а також схеми для програмування та зв'язку з мікроконтролером через USB. Окрім того, контакти введення / виводу мікроконтролера, як правило, вже випливають із сокетів / заголовків для зручного доступу. На стороні програмного забезпечення, Arduino надає ряд бібліотек, що полегшує програмування мікроконтролера. Найпростішими з них є функції для керування та читання контактів вводу-виводу, а не для того, щоб уникнути масок шини / біт, які зазвичай використовуються для взаємодії з ATmega I/O. Arduino - це платформа створення прототипів електроніки з відкритим вихідним кодом, що базується на гнучких, простих у використанні апаратних і програмних продуктах. Arduino складається з фізичної програмованої плати та програмного забезпечення, або IDE (інтегрованого середовища розробки)

Особливості використовуваних мікроконтролерів ATmega фірми Atmel дозволяють виконувати програмування без застосування спеціальних програматорів. Все, що потрібно для створення нового електронного пристрою, це плата Arduino, кабель зв'язку та комп'ютер. Другою частиною проекту Arduino є програмне забезпечення для створення керуючих програм. Воно об'єднало в собі найпростішу середу розробки та мову програмування, що представляє собою варіант мови C / C ++ для мікроконтролерів. У нього додані елементи, що дозволяють створювати програми без вивчення апаратної частини. Так що для роботи з Arduino практично досить знання тільки основ програмування на C / C ++.

На сьогоднішній день існує безліч моделей мікроконтролерів від Arduino. Так як концепція IoT потребує можливості взаємодіяти з іншими пристроями в локальній чи інтернет мережі, то ми не будемо розглядати такі моделі як Nano, Uno чи Mega, які потребують додаткових плат розширень Wi-Fi чи Ethernet, а звернемо увагу на комплексні рішення такі як:

- Arduino Ethernet
- Arduino Yun
- Arduino TIAN

2.1.1 Arduino Ethernet

Arduino Ethernet це фактично Arduino Uno з вбудованою платою розширення Ethernet. Тобто основною відмінністю є присутність RJ – 45 та microSD роз'ємів. Вартість становить приблизно $43. Його зовнішній вигляд наведено на рис 2.1, а технічні характеристики у таблиці 2.1 [4].

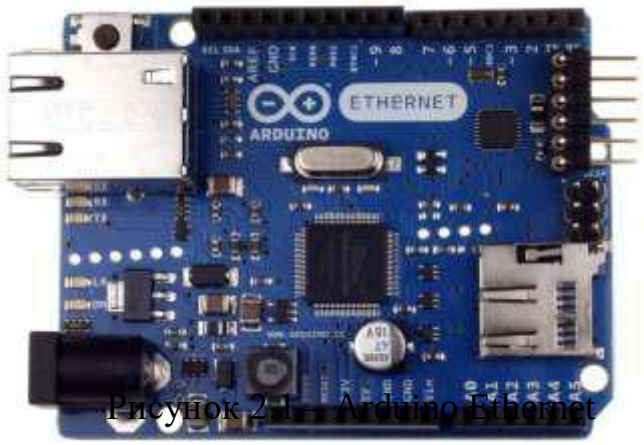

Рисунок 2.1 – Arduino Ethernet

Таблиця 2.1 – Характеристики Arduino Ethernet

Параметр	**Значення**
Мікроконтролер	ATmega328
Робоча напруга	5В
Вхідна напруга (рекомендований)	7-12В
Цифрові Входи / Виходи	14 (4 ШІМ-виходи)
Flash-пам'ять	32 КВ
SRAM	2 КВ
Незалежна пам'ять	1 КВ
Тактова частота	16 MHz
Підключення до локальної мережі	Ethernet

2.1.2 Arduino Yun

Arduino YUN - це перший представник новітньої серії плат Arduino з вбудованим WiFi, що поєднують в собі найширші можливості Linux і простоту

використання Arduino. Arduino YUN є комбінацією класичного Arduino Leonardo (на базі мікроконтролера ATmega32U4) і WiFi-системи на кристалі, що працює під управлінням Linino (дистрибутив ОС GNU / Linux на основі OpenWRT для мікропроцесорів MIPS). Вартість складає близько $69. Зовнішній вигляд наведено на рис 2.2, а технічні характеристики у таблиці 2.2 [5].

Рисунок 2.2 – Arduino YUN

Таблиця 2.2 – Характеристики Arduino YUN

Параметр	**Характеристика**
Мікроконтролер	ATmega32u4
Робоча напруга	5В
Вхідна напруга (рекомендований)	5В чи PoE 802.3af
Цифрові Входи / Виходи	14 (7 ШІМ-виходи)
Flash-пам’ять	32 КВ
SRAM	2.5 КВ
Незалежна пам'ять	1 КВ
Тактова частота	16 MHz
Підключення до локальної мережі	Ethernet, Wi-Fi
Процесор	400 MHz
RAM	64 MB
Flash-память (Процесора)	16 MB

2.1.3 Arduino TIAN

Найпотужніший Arduino, який побудоано на базі 32 бітного процессора з можливістю підключення по Wi-FI 2.4/5 ГГц та Ethernet. Вартість приблизно $92. Його зовнішній вигляд наведено на рис 2.3, а технічні характеристики у таблиці 2.3 [6].

Рисунок 2.3 – Arduino TIAN

Таблиця 2.3 – Характеристики Arduino TIAN

Параметр	**Характеристика**
Мікроконтролер	SAMD21G18
Робоча напруга	3.3В
Вхідна напруга (рекомендований)	5В
Цифрові Входи / Виходи	14 (12 ШІМ-виходи)
Flash-пам’ять	256 КВ
SRAM	32 КВ
Незалежна пам'ять	4 КВ
Тактова частота	48 MHz
Підключення до локальної мережі	Ethernet, Wi-Fi
Процесор	560 MHz
RAM	64 MB
Flash-память (Процесора)	16 MB + 4 GB eMMC

Таким чином сучасні мікроконтролери від Arduino мають досить високу вартість та за характеристиками власне чіпів мікроконтролерів мало відрізняються від попередніх проектів, позитивним є чудова їх підтримка, яка дозволяє початківцям отримати навички з роботи з ними.

2.1.4 Оцінка обчислювальної потужності Arduino

Розглянемо обчислювальні потужності контролерів на базі чіпів Atmega328 та Atmega2560. Для оцінки будемо додавати елементи short int масиву та виконувати get запити. Дані по роботі з масивами наведено у таблицях 2.4, 2.5.

Таблиця 2.4 – Робота з масивом на Atmega 328

Розмір масиву	**Спроба, мкс**					**Середній**
	1	**2**	**3**	**4**	**5**	
500	5416	5320	5408	5488	5516	5429.6
900	9712	9660	9964	9876	10116	9865.6

Таблиця 2.5 – Робота з масивом на Atmega 2560

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	5752	5800	5692	5880	5760	5776.8
1000	12184	11360	11624	12368	12176	11942.4
2500	29752	30164	30480	30056	28556	29801.6
3500	40972	41308	40728	42516	42968	41698.4

В якості тесту мережевого з’єднання перевіримо виконання запитів get сайту www.kpi.ua. Для підключення до мережі інтернет скористаємося платою розширення з чіпом w5100 та зробимо по 5 спроб по 10 запитів у кожній. Дані наведено у таблиці 2.6.

Таблиця 2.6 – Робота Arduino з get

Чіп	**Спроба (10 заптів), мкс**					**Середній, мкс**	**1 запит у середньому, мкс**
	1	**2**	**3**	**4**	**5**		
atmega328	11344144	13172020	13153576	11312408	13159716	12428372.80	1242837.28
atmega2560	13188420	13423952	11393332	13223156	13186344	12883040.80	1288304.08

2.2 Мікроконтролери ESP

Поява мікроконтролерів від Espressif стала наступним суттєвим кроком у розвитку IoT. Якщо продукти від Arduino дозволило кожному бажаючому розпочати програмування мікроконтролерів, то продукти від Espressif дали можливість так само легко об’єднувати їх у локальні мережі за допомогою Wi-Fi, збільшивши при цьому їх потужність та зменшивши собівартість.

2.2.1 ESP-8266

Мікроконтролер ESP-8266 з’явився у 2014 році. Він дозволяє за допомогою вбудованого Wi-Fi підключатися до мереж та оновлювати ПЗ. Існує велика кількість модифікацій, розглянемо останню, яка побудована на чіпі ESP-12 від WEMOS Electronic. Вартість складає $3-5. Його зовнішній вигляд наведено на рис 2.4, а технічні характеристики у таблиці 2.7 [7].

Рисунок 2.4 – WEMOS D1 mini pro

Параметр	Значення
Мікроконтролер	ESP-8266EX
Робоча напруга	3.3В
Вхідна напруга (рекомендований)	5В
Цифрові Входи / Виходи	11
Flash-пам’ять	4/16 MB
Тактова частота	80/120 MHz
Підключення до локальної мережі	Wi-Fi

2.2.2 ESP-32

У 2015 році компанія Espressif представила новий мікроконтролер ESP-32. У ньому було встановлено Bluetooth модуль на додачу до Wi-Fi та процесор з 2 ядрами. Орієнтовна вартість $15-30. Його зовнішній вигляд наведено на рис 2.5, а технічні характеристики у таблиці 2.8 [8].

Рисунок 2.5 – SparkFun ESP32 Thing

Таблиця 2.8 – Характеристики SparkFun ESP32 Thing

Параметр	**Значення**
Мікроконтролер	ESP-32
Робоча напруга	2.2-3.6В
Вхідна напруга (рекомендований)	5В
Цифрові Входи / Виходи	28
Flash-пам’ять	4 MB
Тактова частота	240 MHz, 2 ядра
Підключення до локальної мережі	Wi-Fi

Таким чином, можна вважати що продукти від Espressif є чудовим поєднанням ціни та продуктивності.

2.2.3 Оцінка обчислювальної потужності ESP

Порівняємо роботу ESP з таким самим масивом як і Arduino та get запитами. Для роботи будемо використовувати фреймворк Arduino та середу розробки PlatformIO.

У таблицях 2.9, 2.10, 2.11 наведено дані по роботі з масивами для чіпів ESP8266 на частотах 80МГц та 160МГц та ESP32.

Таблиця 2.9 – Робота ESP8266 80 МГц з масивом

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	743	706	723	703	701	715.2
1000	1468	1469	1408	1441	1396	1436.4
2500	3487	3507	3647	3517	3495	3530.6
10000	14096	14191	14274	14082	14086	14145.8
20000	28002	28271	28571	28325	28293	28292.4

Таблиця 2.10 – Робота ESP8266 160 МГц з масивом

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	369	372	352	350	367	362
1000	717	700	730	703	710	712
2500	1776	1754	1762	1749	1789	1766
10000	7122	7056	7006	7039	6990	7042.6
20000	13986	14178	14095	14071	13952	14056.4

Таблиця 2.11 – Робота ESP32 з масивом

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	239	282	283	295	205	260.8
1000	431	410	439	407	408	419
2500	1061	1030	1046	1035	1054	1045.2
10000	4235	4147	4123	4079	4149	4146.6
20000	8387	8261	8196	8269	8406	8303.8
50000	20786	20864	20368	20325	20320	20532.6

Перевірку за get запитів виконаємо аналогічно Arduino, відмінністю буде підключення до мережі інтернет через вбудований модуль Wi-Fi. Дані наведено у таблиці 2.12.

Таблиця 2.12 – Робота ESP з get запитами

Чіп	Спроба (10 заптів), мкс					Середній, мкс	1 запит у середньому, мкс
	1	2	3	4	5		
esp8266 80mhz	10198784	10094795	10116844	10244791	10131784	10157399.60	1015739.96
esp8266 160mhz	10189839	10176841	10094827	10080844	10094830	10127436.20	1012743.62
esp32	10545813	10386885	10081844	10388835	10436859	10368047.20	1036804.72

2.3 Міні комп’ютери з ОС LINUX

Альтернативою мікроконтролерам при побудові IoT пристроїв можуть виступати міні комп’ютери з ОС Linux. Існує безліч варіантів їх реалізації для використання в IoT необхідна наявність GPIO та бажано компактні розміри. Керуючись з цими міркуваннями розглянемо пристрої від наступних виробників:

- Raspberry Pi
- NanoPi
- Orange Pi

2.3.1 Raspberry Pi

Перші Raspberry Pi з’явилися у 2012 році. Саме ці пристрої поклали основу міні комп’ютерам, які ентузіасти почали використовувати для побудову IoT пристроїв, потужніших за Arduino.

На даний момент найменшим є Arduino Zero. Він не має Ethernet чи Wi-Fi модулів і вимагає їх підключення через OTG. Анонсована вартість складає $5, але

фактично придбати його можливо лише за набагато більшу вартість. Його вигляд наведено на рис 2.6, а технічні характеристики у таблиці 2.13 [9].

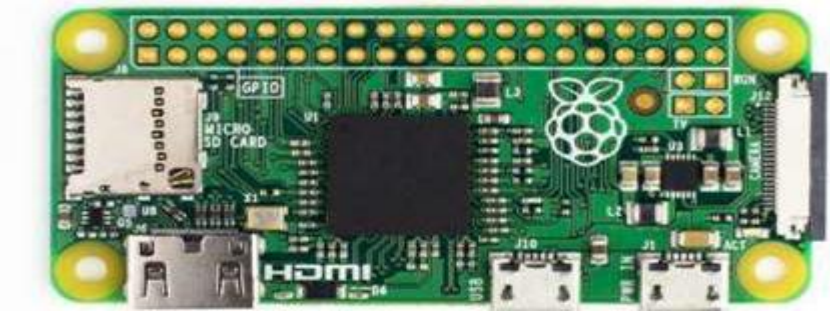

Рисунок 2.6 – Raspberry Pi Zero

Таблиця 2.13 – Характеристики Raspberry Pi Zero

Параметр	**Характеристика**
Процесор	1Ghz, Single-core CPU
Оперативна пам’ять	512 MB
Пам’ять	- (microSD)
Входи / Виходи	40
USB	OTG
Підключення до локальної мережі	- (зовнішні модулі)

Старшою моделлю є Raspberry Pi 3B, вона має 64 бітний процесор, вбудовані Ethernet, Wi-Fi та Bluetooth, а вартість стартує від $40. Зовнішній вигляд наведено на рис 2.7, технічні характеристики у таблиці 2.14 [10].

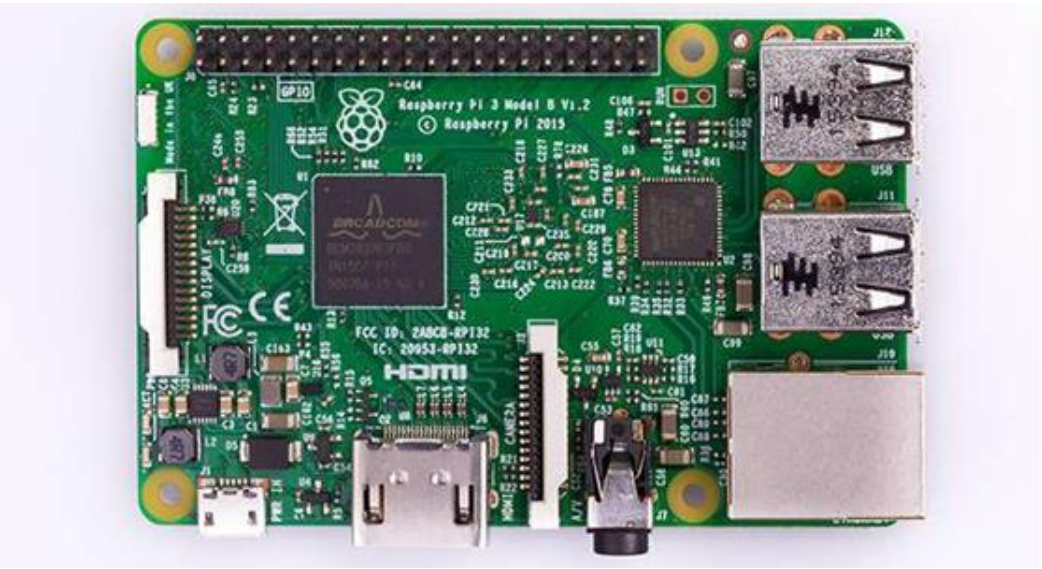

Рисунок 2.7 – Raspberry Pi 3B

Таблиця 2.14 – Характеристики Raspberry 3B

Параметр	Характеристика
Процесор	1.2GHz 64-bit quad-core ARMv8
Оперативна пам’ять	1024 MB
Пам’ять	- (microSD)
Входи / Виходи	40
USB	4
Підключення до локальної мережі	Ethernet, Wi-Fi

Таким чином, Raspberry Pi Zero вимагає додаткових модулів для комунікації, що робить його вартість не дуже привабливою. Raspberry Pi 3B має досить потужний процесор та усі необхідні модулі для взаємодії, але за його вартість можна придбати 10 плат ESP-8266, до того ж його потужність у більшості випадків буде надлишковою. Доцільним мабуть було б його використання у якості центрального вузла систем розумний будинок. Також слід зауважити високий рівень підтримки ПЗ Raspberry Pi, він набагато вище ніж у інших виробників.

2.3.2 NanoPi

Після тріумфу з появи Raspberry Pi вже пройшов час і з’явилися аналоги від інших виробників. Розглянемо деякі моделі NanoPi, які є досить компактними для побудови IoT пристроїв.

NanoPi Neon дуже маленький міні комп’ютер розміром трохи більше коробки сірників. Загалом він більш потужний за Raspberry Pi Zero, та має повноцінний USB та вбудований Ethernet модуль. Вартість складає $10-12. Його вигляд наведено на рис 2.8, технічні характеристики у таблиці 2.15 [11].

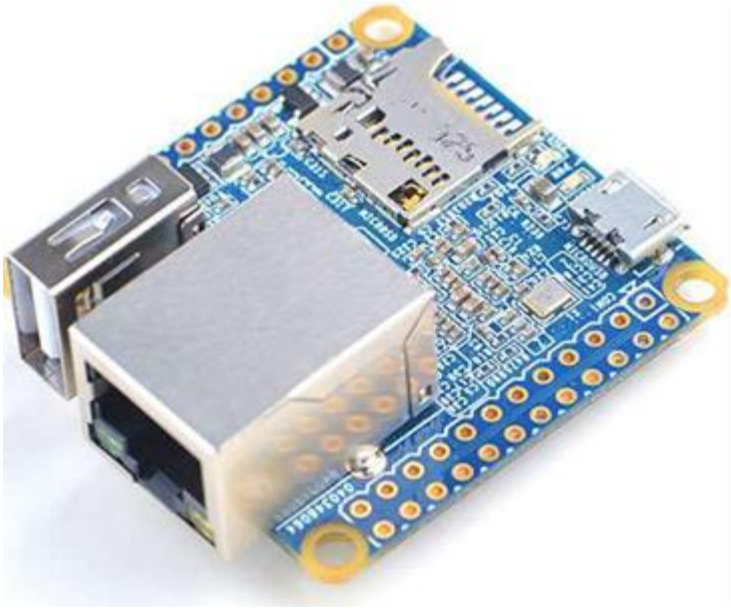

Рисунок 2.8 – NanoPi Neon

Таблиця 2.15 – Характеристики NanoPi Neon

Параметр	**Характеристика**
Процесор	Quad-core Cortex-A7 Up to 1.2GHz
Оперативна пам’ять	256/512 MB
Пам’ять	- (microSD)
Входи / Виходи	36
USB	4
Підключення до локальної мережі	Ethernet

NanoPi-NEO-Air фактично є версією NanoPi Neon з Wi-Fi, без Ethernet та USB з eMMC. Вартість складає $26. Його зовнішній вигляд наведено на рис 2.9, а технічні характеристики у таблиці 2.16 [12].

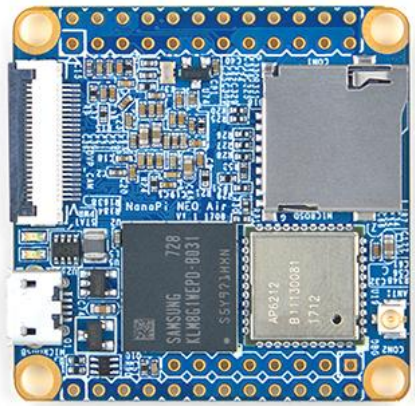

Рисунок 2.9 – NanoPi-NEO-Air

Таблиця 2.16 – Характеристики NanoPi-NEO-Air

Параметр	Характеристика
Процесор	Quad-core Cortex-A7 Up to 1.2GHz
Оперативна пам’ять	512 MB
Пам’ять	8GB eMMC
Входи / Виходи	36
USB	OTG
Підключення до локальної мережі	Wi-Fi

2.3.3 Orange Pi

Orange Pi це інший виробник міні комп’ютерів, які можуть виступати альтернативою Rasberry Pi. Orange Pi Zero це міні комп’ютер розміром трохи більшим коробки сірників. Він має вбудований Ethernet та Wi-Fi модуль. Його вартість складає $7-9. Зовнішній вигляд наведено на рис 2.10, а технічні характеристики у таблиці 2.17 [13].

Рисунок 2.10 – Orange Pi Zero

Таблиця 2.17 – Характеристики Orange Pi Zero

Параметр	**Характеристика**
Процесор	Quad-core Cortex-A7 Up to 1.2GHz
Оперативна пам’ять	256/512 MB
Пам’ять	- (microSD)
Входи / Виходи	26
USB	1
Підключення до локальної мережі	Ethernet, Wi-Fi

2.3.4 Оцінка обчислювальної потужності Linux міні ПК

На міні ПК ми оцінимо роботу з масивом, аналогічним тому, що був на мікроконтролерах та get запитами на мовах програмування:

- C++
- Java

Python

Порівняємо роботу на чіпах:

- BCM2837 (Raspberry Pi 3B)
- Alwinner H2+ (Orange pi Zero)
- BCM2835 900 МГц (Raspberry Pi B+)
- BCM2835 700 МГц (Raspberry Pi B)

Розглянемо можливості Raspberry Pi B. У таблицях 2.18, 2.19, 2.20 наведено результати роботи міні ПК з масивом на мовах C++, Java та Python.

Таблиця 2.18 – Робота Raspberry Pi B з масивом на мові C++

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	61	55	58	59	58	58.2
1000	104	102	100	100	96	100.4
2500	232	227	223	223	226	226.2
10000	1231	863	862	1185	1158	1059.8
20000	1793	2003	1794	1980	1715	1857
50000	4536	4700	4578	4570	4699	4616.6
100000	8846	8925	9018	8985	8973	8949.4
500000	43482	43346	43356	43373	43577	43426.8
1000000	86423	86390	86829	86462	86473	86515.4

Таблиця 2.19 – Робота Raspberry Pi B з масивом на мові Java

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	349	341	342	345	343	344
1000	672	716	666	726	667	689.4
2500	1758	1695	1681	1888	1742	1752.8
10000	7147	7019	7412	7424	7450	7290.4
20000	14288	14624	14101	13138	14269	14084
50000	36388	37181	37003	36811	36935	36863.6
100000	72972	73398	74114	73030	71561	73015
500000	393625	385094	394484	390129	392586	391183.6
1000000	1041554	1057389	1041417	1036211	1032176	1041749.4

Таблиця 2.20 – Робота Raspberry Pi B з масивом на мові Python

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	949	756	765	776	857	820.6
1000	1538	1532	1579	1682	1791	1624.4
2500	3781	4164	3877	3964	3822	3921.6
10000	16859	17043	19399	16545	18203	17609.8
20000	52954	51594	51030	41594	66952	52824.8
50000	91659	109398	91459	90147	104782	97489
100000	223719	184523	184428	190008	197432	196022
500000	927385	1169856	910268	943585	943781	978975
1000000	1758225	1793586	2007110	2016870	1820199	1879198

Графік порівняння часу роботи з масивом наведено на рис. 2.11.

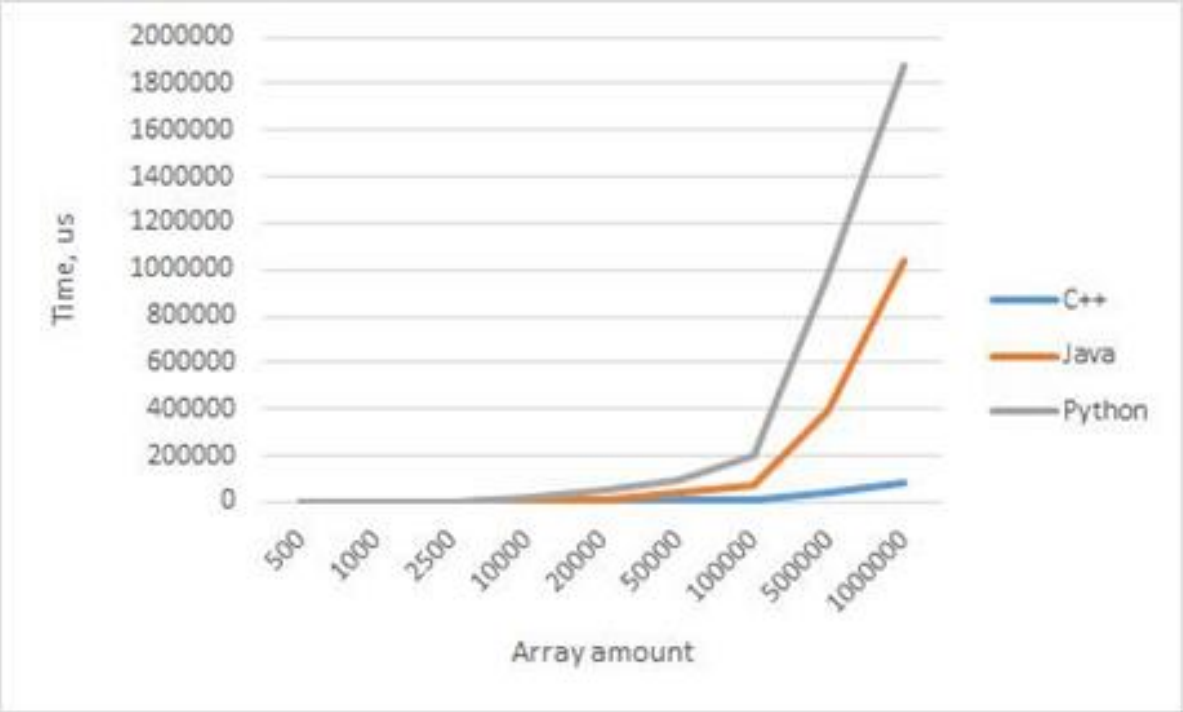

Рисунок 2.11 – Порівняння часу роботи з масивом на Raspberry Pi B

У таблиці 2.21 наведено результати роботи міні ПК Raspberry Pi B з get запитом на мовах C++, Java та Python. Тут слід зауважити, що так як у C++ відсутній стандартній метод, то виконання команді get тут і для всіх інших мікроконтролерів буде реалізовано через виклик стандартного методу wget.

Таблиця 2.21 – Робота Raspberry Pi B з get

	C++	**Java**	**Python**
Спроба 1, мкс	8868	32523321	107062913
Спроба 2, мкс	8992	32138767	106868502
Спроба 3, мкс	7715	32223424	106677047
Спроба 4, мкс	8408	32514622	115084619
Спроба 5, мкс	8114	31958117	128274945
Середній, мкс	8419.4	32271650.2	112793605.2
Середній час на 1 запит, мкс	841.94	3227165.02	11279360.52

Діаграму порівняння часу, необхідного для get запиту, наведено на рис. 2.12.

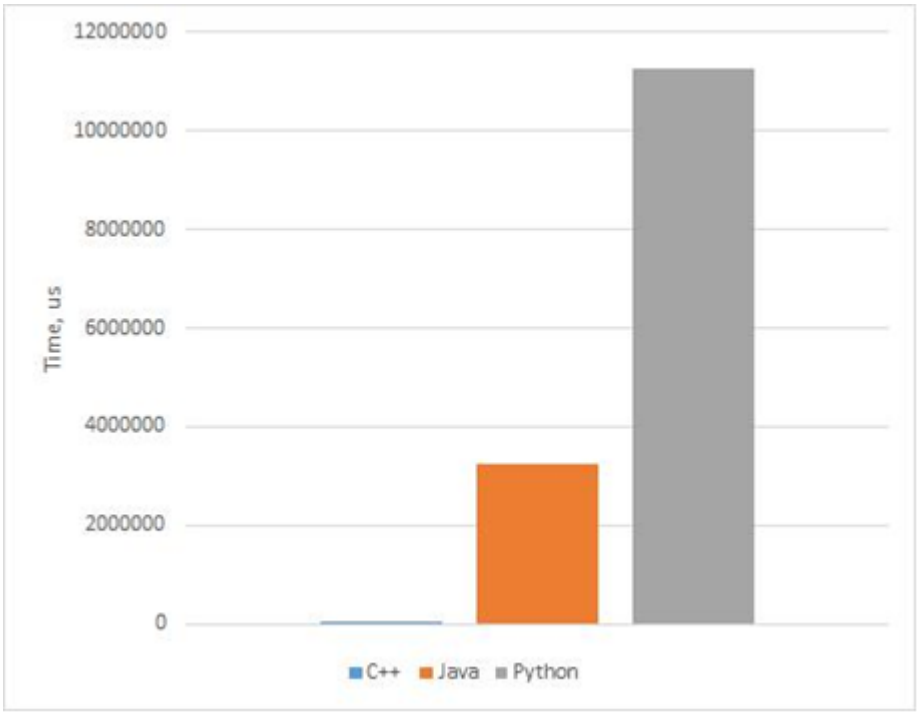

Рисунок 2.12 – Порівняння часу get запиту на Raspberry Pi B

Тепер розглянемо можливості Raspberry Pi B+ на частоті 900 МГц. У таблицях 2.22, 2.23, 2.24 наведено результати роботи міні ПК з масивом на мовах C++, Java та Python.

Таблиця 2.22 – Робота Raspberry Pi B+ з масивом на мові C++

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	61	56	59	59	60	59
1000	102	99	100	144	101	109.2
2500	234	228	229	228	226	229
10000	868	862	859	863	859	862.2
20000	1710	1706	1706	1985	1992	1819.8
50000	4632	4613	4407	4245	4591	4497.6
100000	8993	8889	8896	8983	8997	8951.6
500000	43346	43379	43404	43428	43405	43392.4
1000000	86425	86528	86295	86507	86253	86401.6

Таблиця 2.23 – Робота Raspberry Pi B+ з масивом на мові Java

Розмір масиву	Спроба, мкс					Середній, мкс
	1	2	3	4	5	
500	346	340	346	339	343	342.8
1000	671	667	665	673	666	668.4
2500	1710	1703	1696	1769	1931	1761.8
10000	7146	7018	7155	7054	7126	7099.8
20000	14159	14120	14281	14432	14119	14222.2
50000	36459	36311	36284	37205	36626	36577
100000	75076	76588	74725	75924	75547	75572
500000	384387	385139	383493	383468	377301	382757.6
1000000	777872	778116	777440	762203	761295	771385.2

Таблиця 2.24 – Робота Raspberry Pi B+ з масивом на мові Python

Розмір масиву	Спроба, мкс					Середній, мкс
	1	2	3	4	5	
500	753	773	807	880	911	824.8
1000	1550	1505	1561	1696	1608	1584
2500	4283	3879	3911	3986	3975	4006.8
10000	17992	16464	17359	17078	17750	17328.6
20000	51051	45406	45257	63766	50159	51127.8
50000	84406	84684	86549	98244	86256	88027.8
100000	185350	176530	188459	191624	185390	185470.6
500000	924998	919485	986265	918937	1140068	977950.6
1000000	1764916	1741495	1835357	1802151	1853143	1799412.4

Графік порівняння часу роботи з масивом наведено на рис. 2.13.

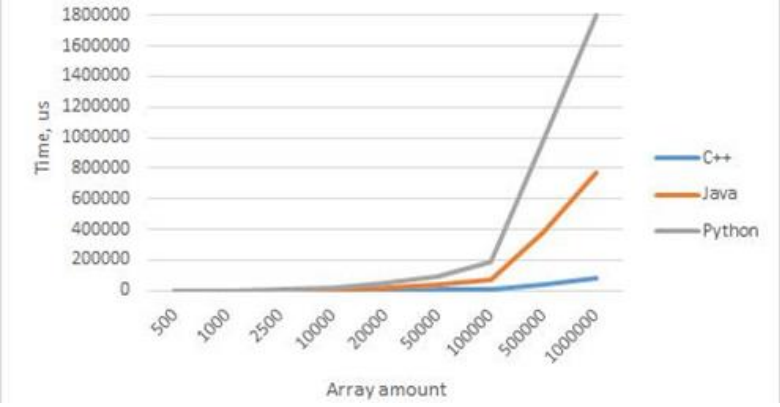

Рисунок 2.13 – Порівняння часу роботи з масивом на Raspberry Pi B+

У таблиці 2.24 наведено результати роботи міні ПК Raspberry Pi B+ з get запитом на мовах C++, Java та Python.

Таблиця 2.24 – Робота Raspberry Pi B+ з get запитами

	C++	**Java**	**Python**
Спроба 1, мкс	7415	31598980	106762354
Спроба 2, мкс	7659	31188980	106781940
Спроба 3, мкс	7179	31366150	106792260
Спроба 4, мкс	7344	31844274	106690711
Спроба 5, мкс	8266	31357745	106879304
Середній, мкс	7572.6	31471225.8	106781313.8
Середній час на 1 запит, мкс	757.26	3147122.58	10678131.38

Діаграма порівняння часу необхідного для get запиту наведено на рис. 2.14.

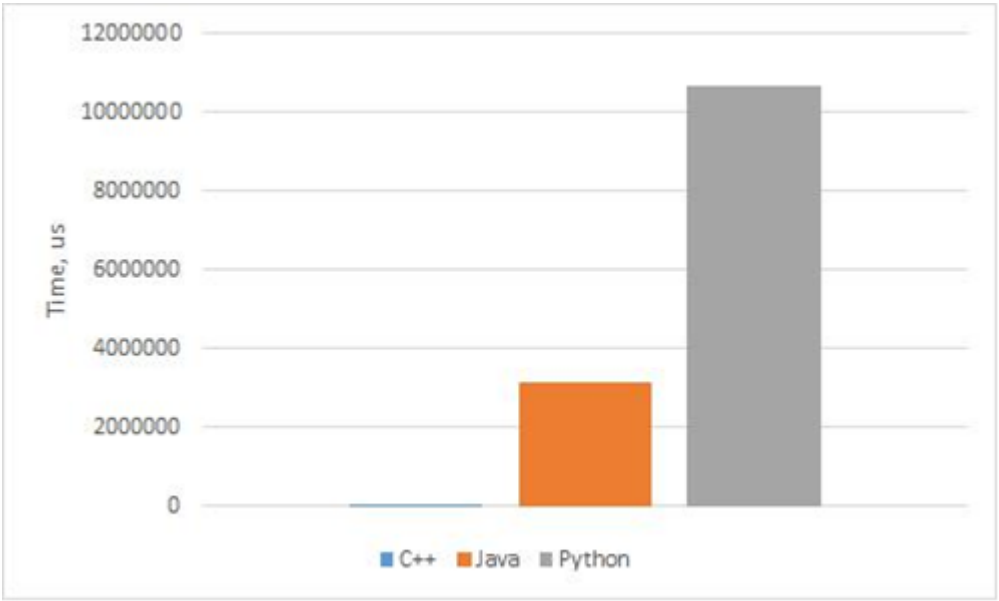

Рисунок 2.14 – Порівняння часу get запиту на Raspberry Pi B

Розглянемо можливості Orange Pi Zero. У таблицях 2.25, 2.26, 2. 27 наведено результати роботи міні ПК з масивом на мовах C++, Java та Python.

Таблиця 2.25 – Робота Orange Pi Zero з масивом на мові C++

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	38	41	37	38	38	38.4
1000	73	72	72	71	70	71.6
2500	172	169	168	168	168	169
10000	662	657	657	681	660	663.4
20000	1314	1347	1341	1310	1307	1323.8
50000	3321	3264	3288	3299	3264	3287.2
100000	6629	6600	6594	6553	6764	6628
500000	33580	33560	33692	33646	35336	33962.8
1000000	67263	67389	67374	67282	67390	67339.6

Таблиця 2.26 – Робота Orange Pi Zero з масивом на мові Java

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	392	392	392	392	391	391.8
1000	785	773	860	773	774	793
2500	1928	1926	1929	1927	1928	1927.6
10000	7839	7851	7775	7842	7772	7815.8
20000	15720	15686	15905	15797	15547	15731
50000	39041	39376	39107	39038	39243	39161
100000	78647	78064	78935	78632	78295	78514.6
500000	394295	393729	395513	394548	395018	394620.6
1000000	790311	789634	789417	789163	791445	789994

Таблиця 2.27 – Робота Orange Pi Zero з масивом на мові Python

Розмір масиву	Спроба, мкс					Середній, мкс
	1	2	3	4	5	
500	884	926	1137	1118	943	1001.6
1000	2175	2184	2200	2235	2184	2195.6
2500	5384	5439	5349	5540	5338	5410
10000	17895	21570	18355	22212	21840	20374.4
20000	37152	44569	37860	46051	45711	42268.6
50000	106288	103782	115603	110312	115053	110207.6
100000	194632	213484	210927	217606	236230	214575.8
500000	1179930	1123158	1170509	1176582	1176763	1165388.4
1000000	1962021	2201430	2312172	2162072	2212021	2169943.2

Графік порівняння часу роботи з масивом наведено на рис. 2.15.

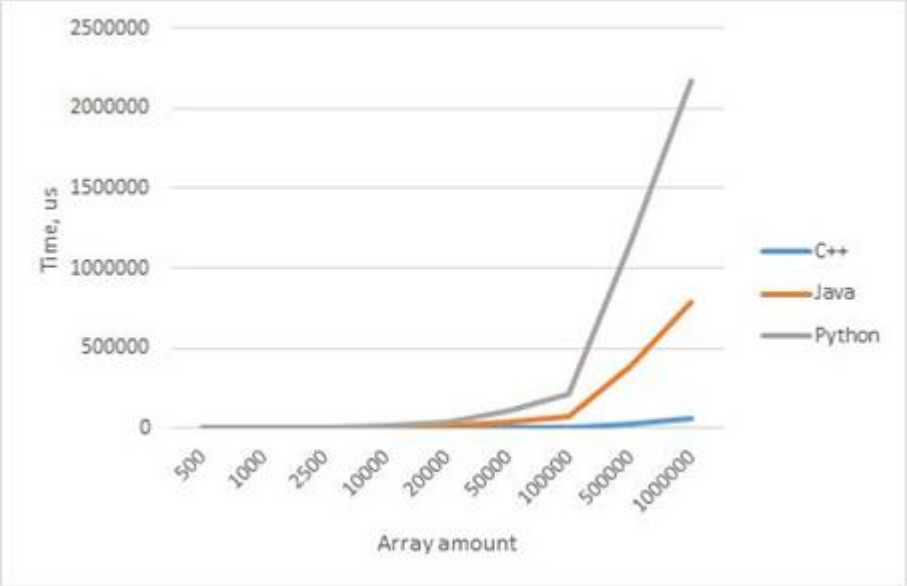

Рисунок 2.15 – Порівняння часу роботи з масивом на Orange Pi Zero

У таблиці 2.28 наведено результати роботи міні ПК Orange Pi Zero з get запитом на мовах C++, Java та Python.

Таблиця 2.28 – Робота Orange Pi Zero з get запитами

	C++	Java	Python
Спроба 1, мкс	6959	23057360	68104533
Спроба 2, мкс	6562	23107796	65714322
Спроба 3, мкс	7131	22806219	66415161
Спроба 4, мкс	6957	22813429	61833606
Спроба 5, мкс	7018	22802202	74184355
Середній, мкс	6925.4	22917401.2	67250395.4
Середній час на 1 запит, мкс	692.54	2291740.12	6725039.54

Діаграму порівняння часу необхідного для get запиту наведено на рис. 2. 16.

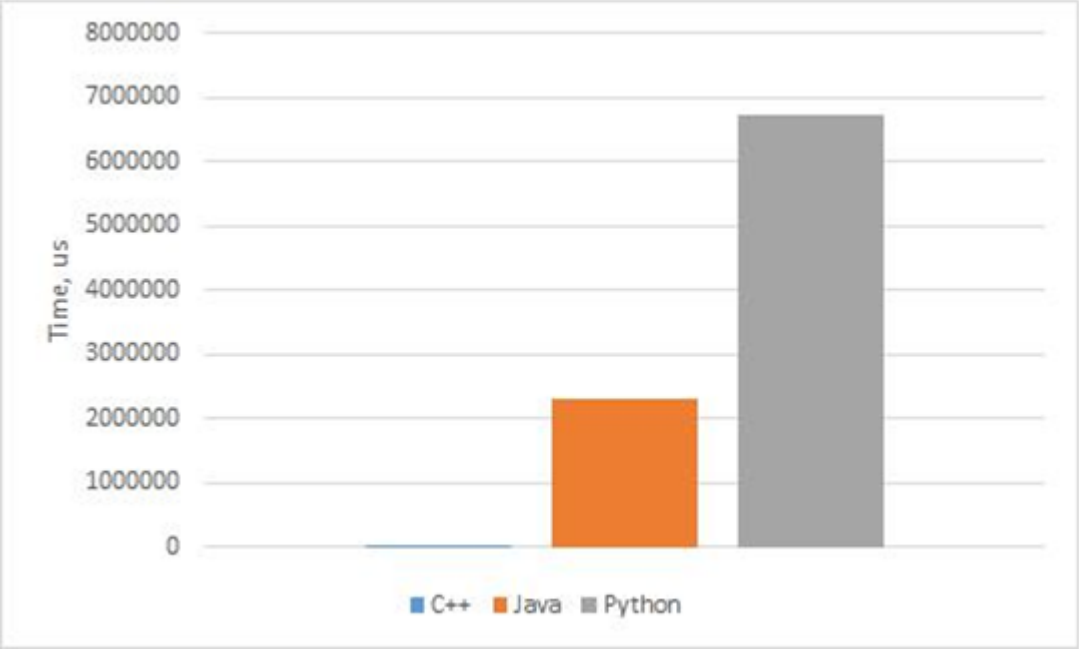

Рисунок 2.16 – Порівняння часу get запиту на Orange Pi Zero

Розглянемо можливості Raspberry Pi 3B. У таблицях 2. 29, 2.30, 2.31 наведено результати роботи міні ПК з масивом на мовах C++, Java та Python.

Таблиця 2.29 – Робота Raspberry Pi 3B з масивом на мові C++

Розмір масиву	Спроба, мкс					Середній, мкс
	1	2	3	4	5	
500	31	29	28	29	28	29
1000	56	55	54	54	54	54.6
2500	133	131	129	132	131	131.2
10000	510	512	511	512	511	511.2
20000	1015	1006	1016	1005	1005	1009.4
50000	2562	2533	2551	2531	2534	2542.2
100000	5063	5052	5125	5045	5078	5072.6
500000	25302	25474	25422	25440	25427	25413
1000000	50611	50329	50251	50393	50296	50376

Таблиця 2.30 – Робота Raspberry Pi 3B з масивом на мові Java

Розмір масиву	Спроба, мкс					Середній, мкс
	1	2	3	4	5	
500	296	194	295	287	248	264
1000	590	583	586	579	583	584.2
2500	744	737	735	737	749	740.4
10000	2978	2906	2969	2973	2841	2933.4
20000	5991	2790	5895	6010	5833	5303.8
50000	29715	30234	29228	24777	23352	27461.2
100000	60435	60413	60638	58489	58045	59604
500000	291306	293589	299809	291765	291213	293536.4
1000000	583388	501184	583950	598401	588926	571169.8

Таблиця 2.31 – Робота Raspberry Pi 3B з масивом на мові Python

Розмір масиву	Спроба, мкс					Середній, мкс
	1	2	3	4	5	
500	388	411	407	462	411	415.8
1000	772	463	463	894	794	677.2
2500	1928	1004	2193	2215	2199	1907.8
10000	8946	8880	7883	8918	8009	8527.2
20000	18205	18597	19148	18034	16068	18010.4
50000	42322	43337	47294	43026	46712	44538.2
100000	82474	48641	93641	48657	89272	72537
500000	214953	371900	224796	361249	230509	280681.4
1000000	831492	765522	470556	894129	941255	780590.8

Графік порівняння часу роботи з масивом наведено на рис. 2.17.

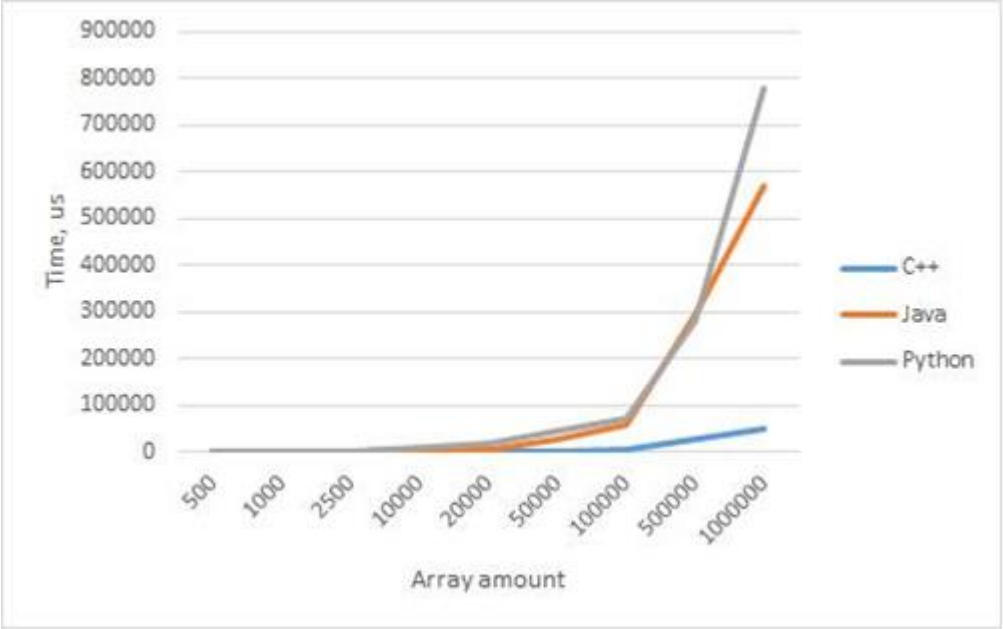

Рисунок 2.17 – Порівняння часу роботи з масивом на Raspberry Pi 3B

У таблиці 2.32 наведено результати роботи міні ПК Raspberry Pi 3B з get запитом на мовах C++, Java та Python.

Таблиця 2.32 – Робота Raspberry Pi 3B з get запитами

	C++	**Java**	**Python**
Спроба 1, мкс	3620	18259217	106498854
Спроба 2, мкс	3711	17480077	106656274
Спроба 3, мкс	3638	17428310	106523145
Спроба 4, мкс	3543	17177134	106621022
Спроба 5, мкс	3438	17625854	106525526
Середній, мкс	3590	17594118.4	106564964.2
Середній час на 1 запит, мкс	359	1759411.84	10656496.42

Діаграму порівняння часу необхідного для get запиту наведено на рис. 2.18.

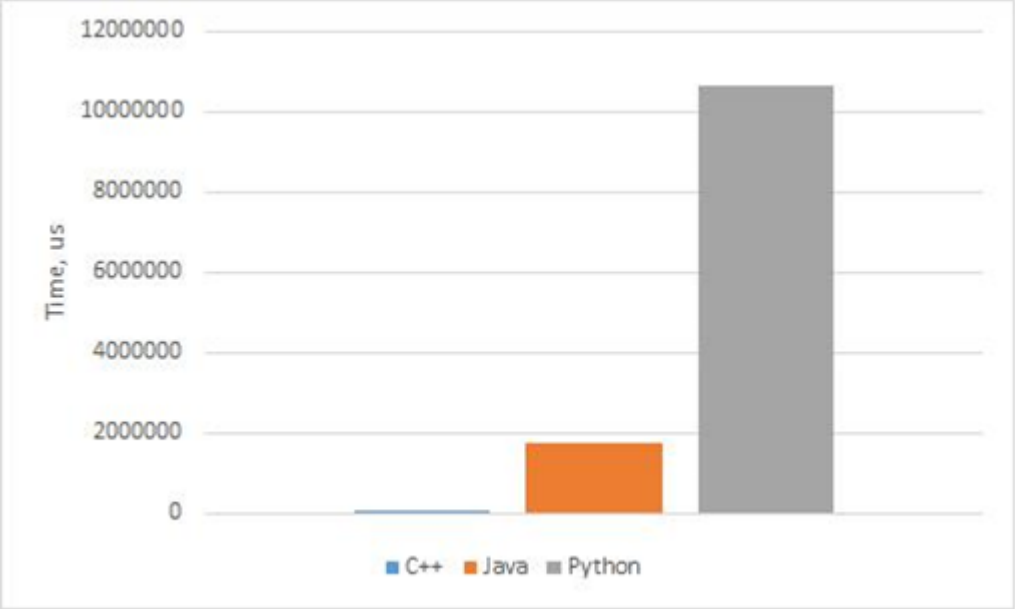

Рисунок 2.18 – Порівняння часу get запиту на Raspberry Pi 3B

2.4 Порівняння обчислювальної потужності

Порівняємо результати тестування для різник міні ПК та мікроконтролерів на мовах C++, Java та Python.

2.4.1 Мова C++

Результати порівняння часу роботи з масивом на мові C++ наведено у таблиці 2.33.

Таблиця 2.33 – Порівняння часу роботи з масивом

Розмір масиву	Nano	Mega 2560	ESP8266 80МГц	ESP8266 160МГц	ESP32	RaspPI 1B	RaspPI 1B+	OrangPi Zero	RaspPI 3B
500	5430	5777	715	362	261	58	59	38	29
900	9866								
1000		11942	1436	712	419	100	109	72	55
2500		29802	3531	1766	1045	226	229	169	131
3500		41698							
10000			14146	7043	4147	1060	862	663	511
20000			28292	14056	8304	1857	1820	1324	1009
50000					20533	4617	4498	3287	2542
100000						8949	8952	6628	5073
500000						43427	43392	33963	25413
1000000						86515	86402	67340	50376

Графік порівняння часу роботи усіх пристроїв з масивом наведено на рис. 2.19, міні ПК – рис. 2.20.

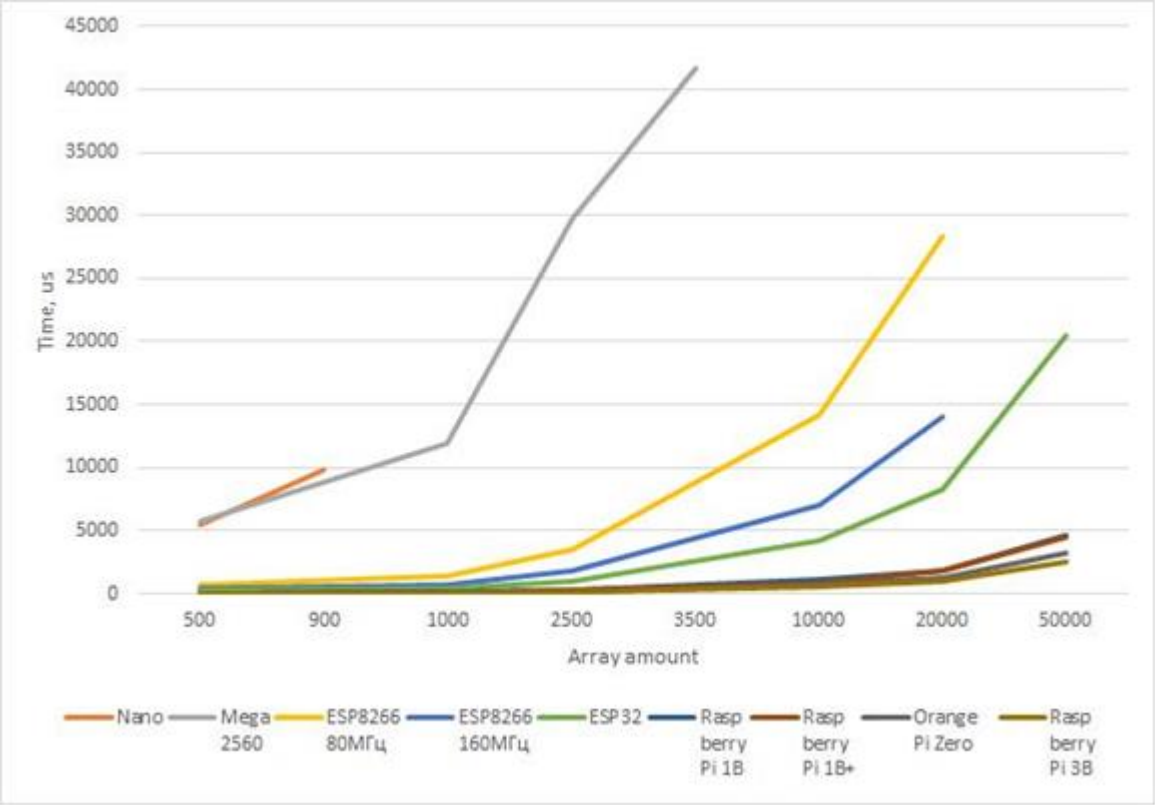

Рисунок 2.19 – Порівняння часу роботи усіх пристроїв з масивом

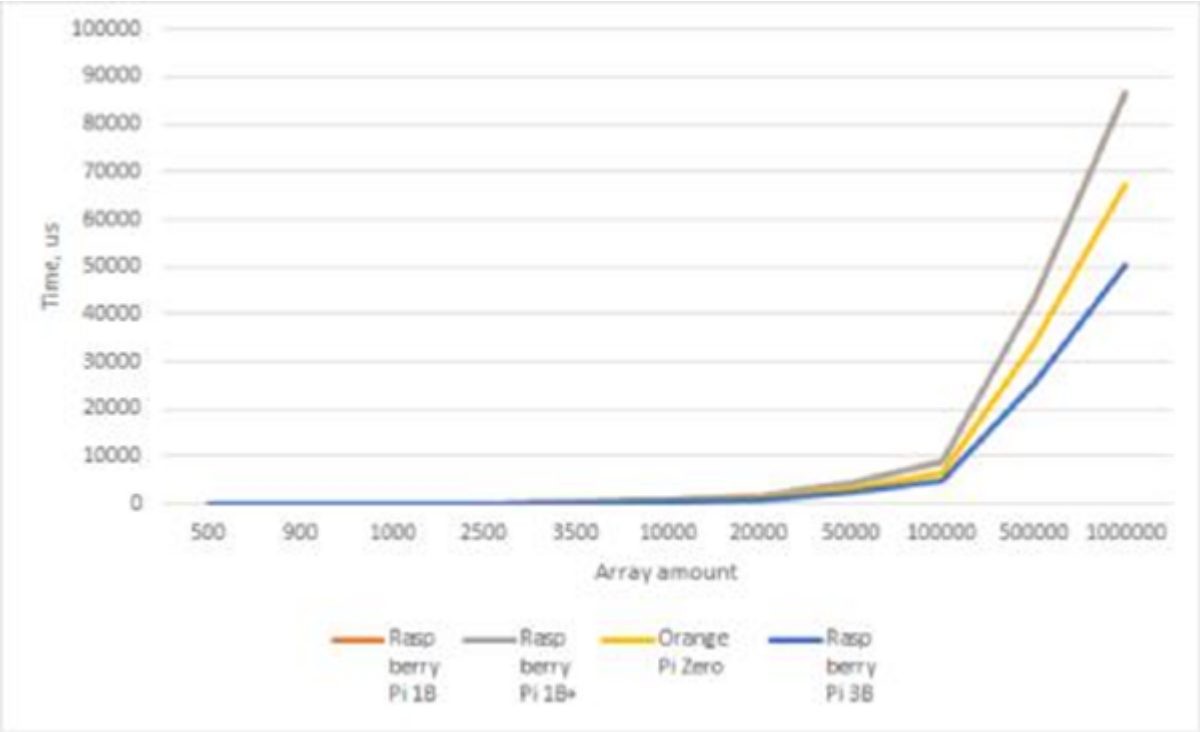

Рисунок 2.20 – Порівняння часу роботи міні ПК з масивом

Результати порівняння часу роботи з get запитом наведено у таблиці 2.34.

Таблиця 2.34 – Порівняння часу роботи з get запитом

Плата	**1 запит у середньому, мкс**
Nano	1242837.28
Mega 2560	1288304.08
ESP8266 80МГц	1015739.96
ESP8266 160МГц	1012743.62
ESP32	1036804.72
Raspberry Pi 1B	841.94
Raspberry Pi 1 B+	757.26
Orange Pi Zero	692.54
Raspberry Pi 3B	359

Діаграми порівняння часу роботи мікроконтролерів з get запитом наведено на рис. 2.21, міні ПК – рис. 2.22.

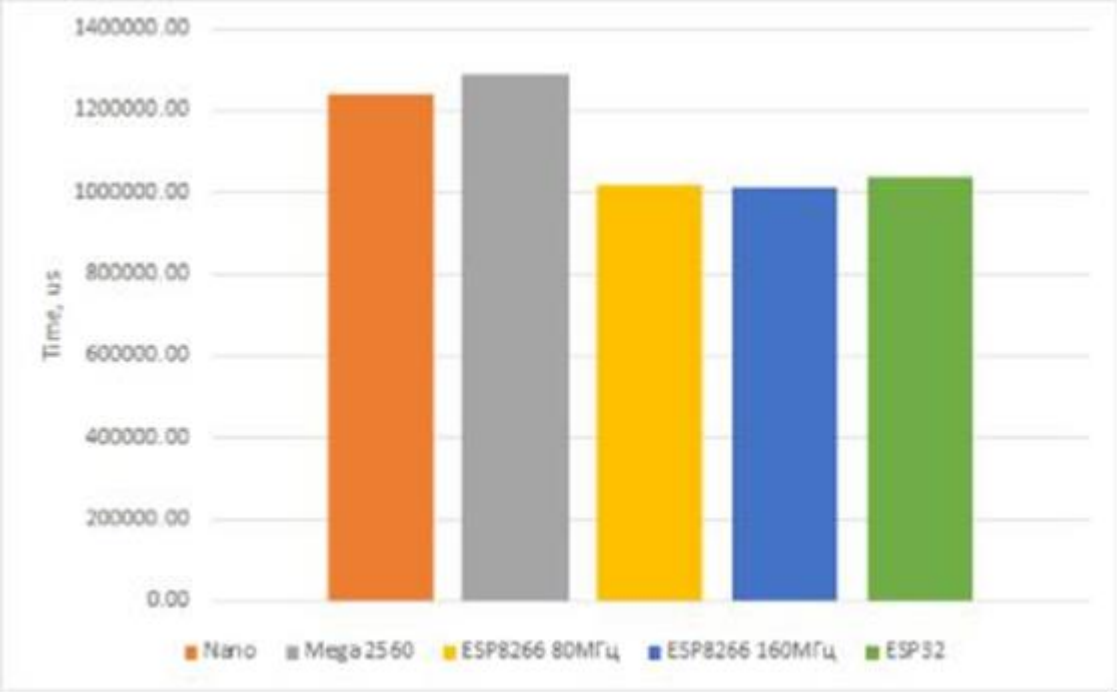

Рисунок 2.21 – Порівняння часу роботи мікроконтролерів з get запитом

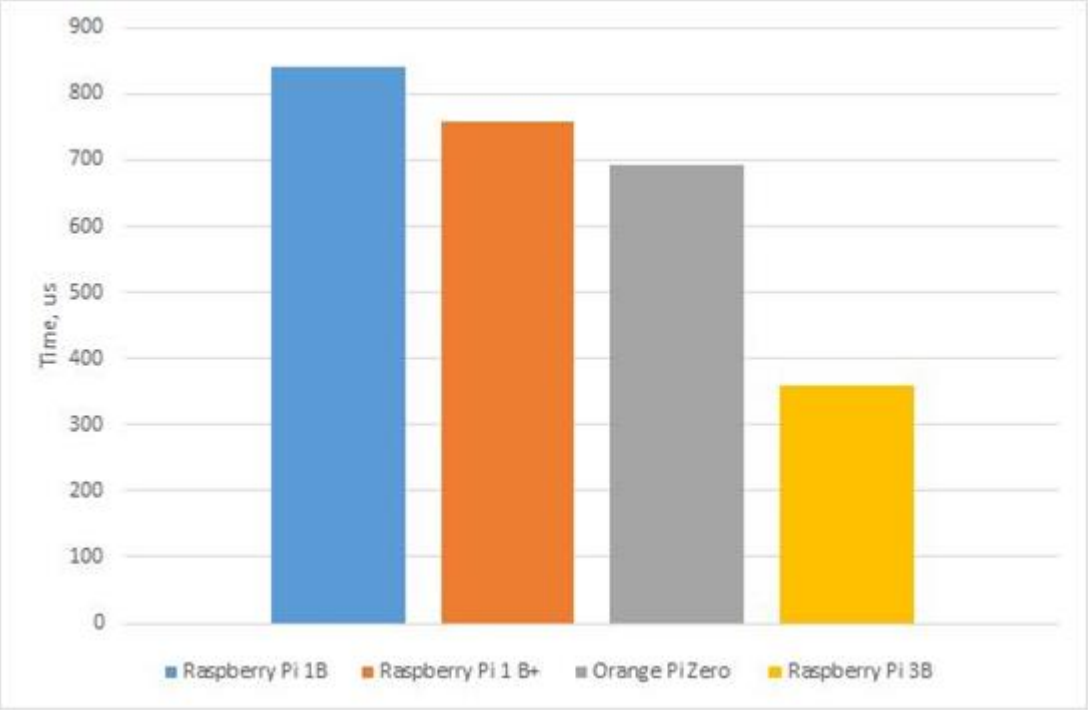

Рисунок 2.22 – Порівняння часу роботи міні ПК з get запитом

2.4.2 Мова Java

Аналогічно попередньому пункту порівняємо час для роботи з масивом та get запитів для міні ПК. У таблиці 2.35 наведено порівняння часу роботи з масивом.

Таблиця 2.35 – Порівняння часу роботи з масивом для міні ПК

Розмір масиву	**Raspberry Pi 1B**	**Raspberry Pi 1 B+**	**Orange Pi Zero**	**Raspberry Pi 3B**
500	344	342.8	391.8	264
1000	689.4	668.4	793	584.2
2500	1752.8	1761.8	1927.6	740.4
10000	7290.4	7099.8	7815.8	2933.4
20000	14084	14222.2	15731	5303.8
50000	36863.6	36577	39161	27461.2
100000	73015	75572	78514.6	59604
500000	391183.6	382757.6	394620.6	293536.4
1000000	1041749.4	771385.2	789994	571169.8

Графік порівняння часу роботи міні ПК з масивом наведено на рис. 2.23

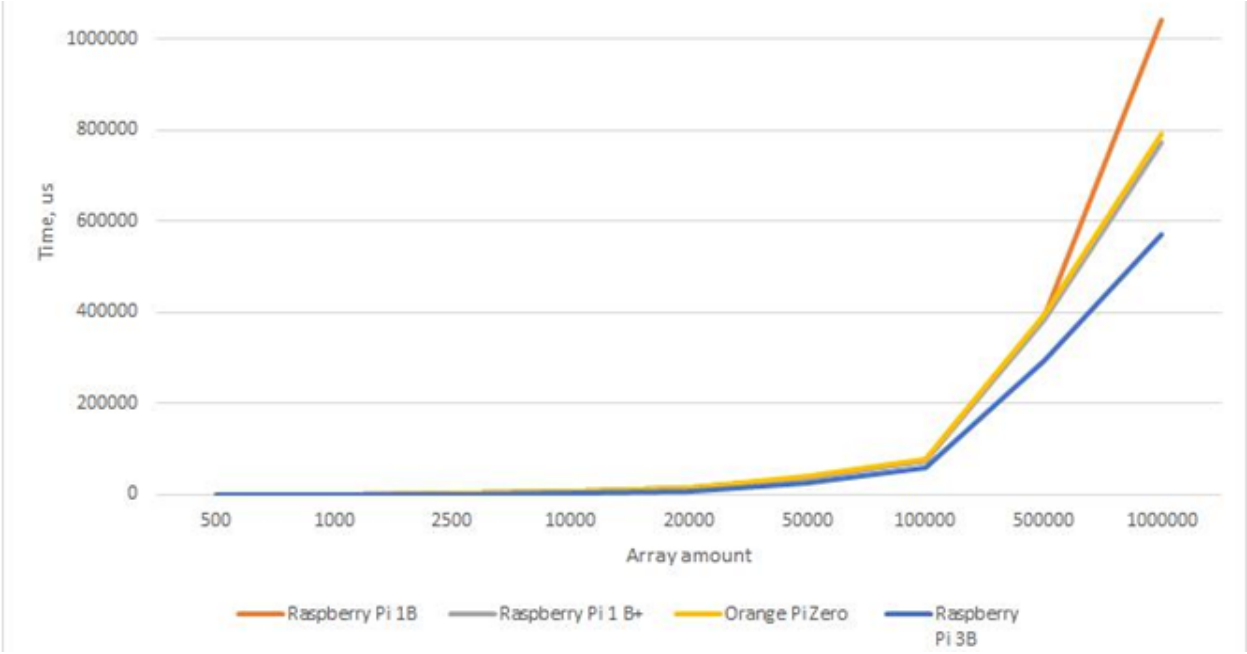

Рисунок 2.23 – Порівняння часу роботи міні ПК з масивом

Результати порівняння часу роботи з get запитом наведено у таблиці 2.36.

Таблиця 2.36 – Порівняння часу роботи з get запитом

	Raspberry Pi 1B	**Raspberry Pi 1 B+**	**Orange Pi Zero**	**Raspberry Pi 3B**
Середній час на 1 запит, мкс	3227165.02	3147122.58	2291740.12	1759411.84

Діаграму порівняння часу роботи міні ПК з get запитом наведено на рис. 2.24

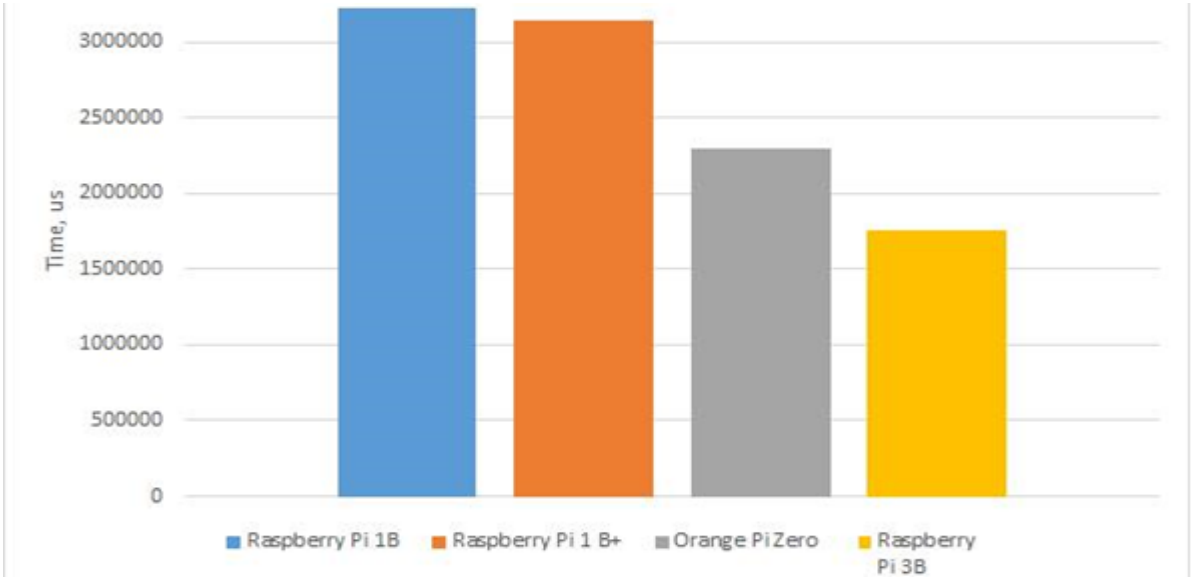

Рисунок 2.24 – Порівняння часу роботи міні ПК з get запитом

2.4.3 Мова Python

Аналогічно попередньому пункту порівняємо час для роботи з масивом та get запитів для міні ПК. У таблиці 2.37 наведено порівняння часу роботи з масивом.

Таблиця 2.37 – Порівняння часу роботи з масивом для міні ПК

Розмір масиву	**Raspberry Pi 1B**	**Raspberry Pi 1 B+**	**Orange Pi Zero**	**Raspberry Pi 3B**
500	820.6	824.8	1001.6	415.8
1000	1624.4	1584	2195.6	677.2
2500	3921.6	4006.8	5410	1907.8
10000	17609.8	17328.6	20374.4	8527.2
20000	52824.8	51127.8	42268.6	18010.4
50000	97489	88027.8	110207.6	44538.2
100000	196022	185470.6	214575.8	72537
500000	978975	977950.6	1165388	280681.4
1000000	1879198	1799412.4	2169943	780590.8

Графік порівняння часу роботи міні ПК з масивом наведено на рис. 2.25

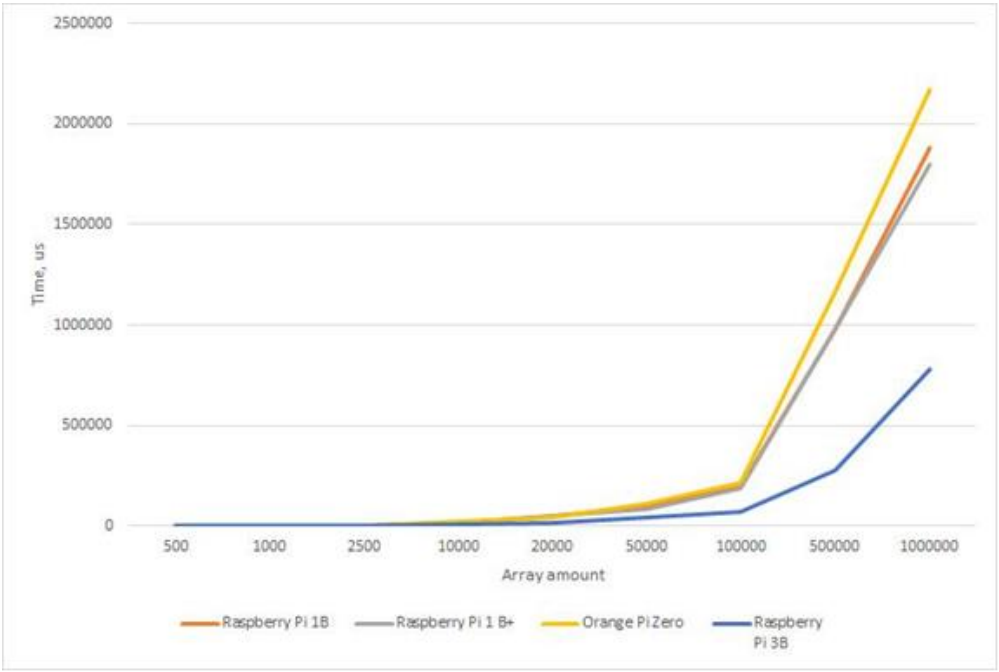

Рисунок 2.25 – Порівняння часу роботи міні ПК з масивом

Результати порівняння часу роботи з get запитом наведено у таблиці 2.38.

Таблиця 2.38 – Порівняння часу роботи з get запитом

	Raspberry Pi 1B	**Raspberry Pi 1 B+**	**Orange Pi Zero**	**Raspberry Pi 3B**
Середній час на 1 запит, мкс	11279360.5	10678131.4	6725039.54	10656496.4

Діаграму порівняння часу роботи міні ПК з get запитом наведено на рис. 2.26

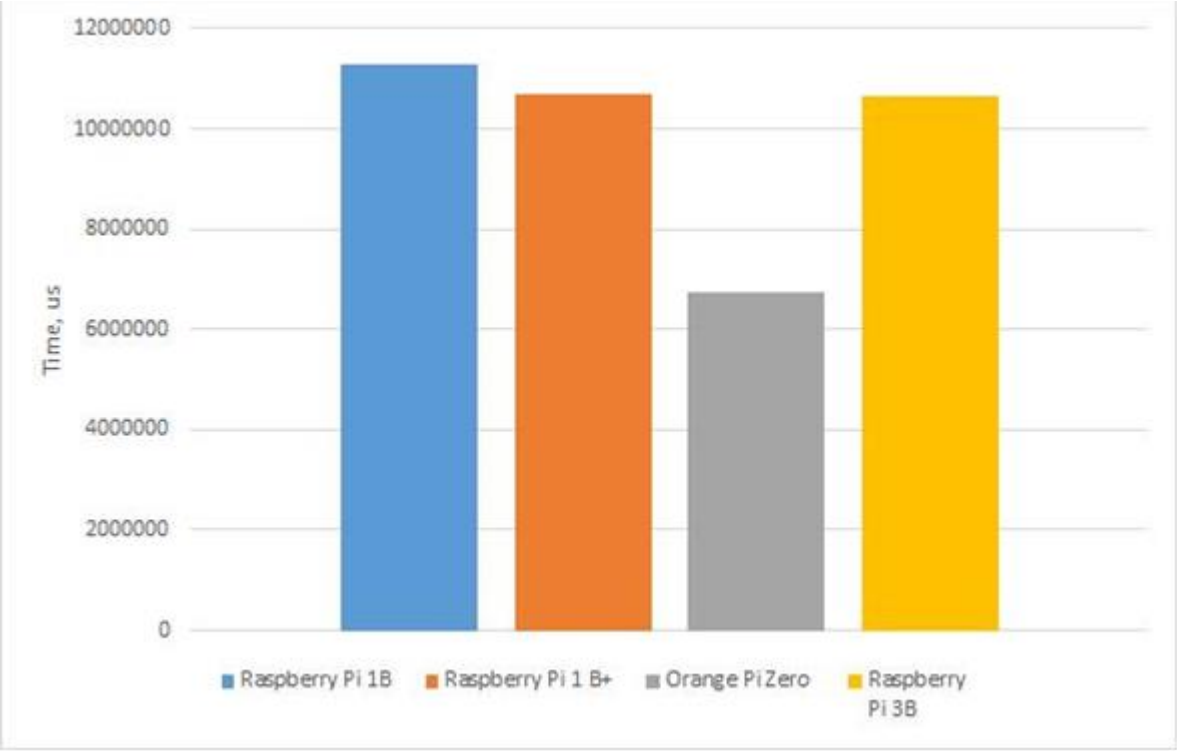

Рисунок 2.26 – Порівняння часу роботи міні ПК з get запитом

2.4.4 Висновки

Як показали тестування, рішення від Arduino мають найменшу ефективність роботи, до того ж їх ціна значно вище ніж у модолей ESP, а також вони не мають базової реалізації підключення до мережі, що робить їх використання недоцільним для IoT пристроїв. ESP32 на відміну від ESP8266 має більшу функціональність, але в нього присутні деякі проблеми з фреймворком, наприклад в Arduino фреймворку відсутня підтримка вбудованої пам’яті, та й ціна поки що на них вища ніж на ESP8266. Таким чином можна

прийти до висновку, що серед мікроконтролерів ідеальним варіантом є ESP8266, адже він має вбудований Wi-Fi, внутрішню пам’ять, досить стабільний фреймворк, та помірну ціну, а також компактні розміри.

Серед міні ПК Raspberry Pi B та B+ показали найнижчі результати, так як Raspberry Pi zero, який побудований на тій самій платформі, то очевидно що його результати будуть схожі. Їх цілком достатньо для простого пристрою, але його собівартість та складність налаштування робить його використання недоцільним ніяк hub пристрою, ні як звичайного пристрою. Оптимальним же для hub пристрою може бути або рішення від Orange Pi або Neo Pi, вони мають близькі характеристики до Raspberry Pi 3B, а але нижчу ціну або вбудовану пам’ять, що робить їх використання більш прийнятним.

2.5 Апаратні модулі необхідні для розробки системи

Для реалізації усіх функцій системи необхідно використати годинник реального часу, джерело живлення, світлодіодну матрицю. Їх характеристики наведені у цьому розділі.

2.5.1 Джерело живлення

Для живлення WiFi модулю від мережевої напруги було використано малогабаритне мережеве джерело живлення на 3.3 (В) TSP-03 (рис 2.27)

Рисунок 2.27 – Джерело живлення

Характеристики:

- вхідна напруга: від 100 до 240В;
- тип вхідної напруги: змінна;
- вихідна напруга: 3,3 В;
- максимальний вихідний струм: 0,9 А;

максимальна вихідна потужність: 3 Вт;

- розміри: 34 х 20 х 15 мм

2.5.2 Модуль реального часу

Для фіксації поточного часу необхідно використати модуль реального часу. Модуль годинника реального часу на базі DS3231SN (рис 2.28). Ця схема отримала свою розповсюдженість завдяки дуже високої точності ходу годинника. Цього вдалося домогтися помістивши кварцовий резонатор в корпус мікросхеми і забезпечивши температурну компенсацію і цифрову корекцію частоти генератора. Внутрішній датчик температури доступний через внутрішні регістри годинника.

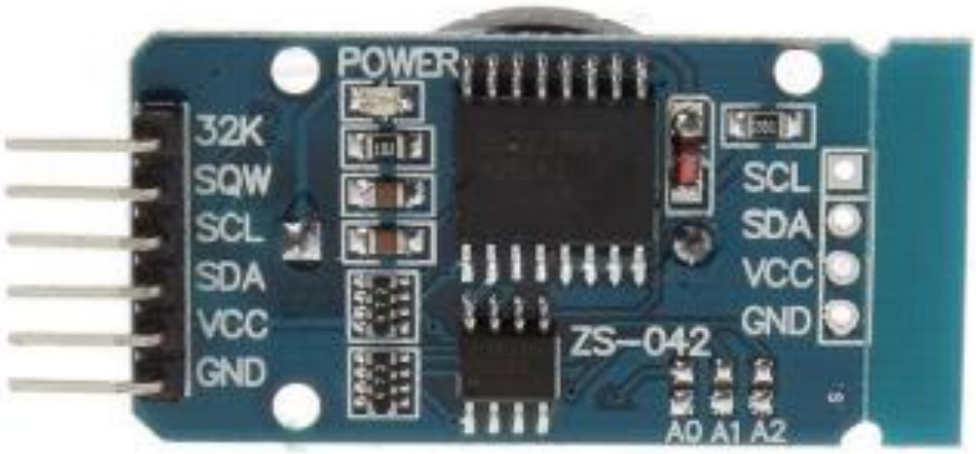

Рисунок 2.28 – модуль годинника

Характеристики:

- висока точність годинного генератора з термокомпенсацією і корекцією ходу;
- лічильники секунд, хвилин, годин, днів тижня, днів, місяців і років з календарем з корекцією високосного року до 2100 року;

- точність внутрішнього цифрового датчика температури ± 3 ° C;
- регістр корекції точності ходу годинника;
- простий і поширений інтерфейс підключення;
- два режими шини I2C: Стандартний (100кГц) і Бистрий (400 кГц);
- батарея резервного живлення годинника;
- робоча напруга живлення від 3.0В до 5.5В.

2.5.3 Світлодіодна матриця

Для візуального відображення інформації була обрана RGB світлодіодна матриця 16x32 фірми Adafruit (рисунок 2.29). Дисплей має 512 яскравих світлодіодів RGB, розташованих в матриці 16x32. На зворотному боці є друкована плата з двома роз'ємами IDC (один вхід, один вихід) і 12 16-бітових регістрів, які дозволяють керувати дисплеєм зі швидкістю сканування 1/8 секунди. Ці дисплеї можна каскадувати - підключити один вихід до наступного входу.

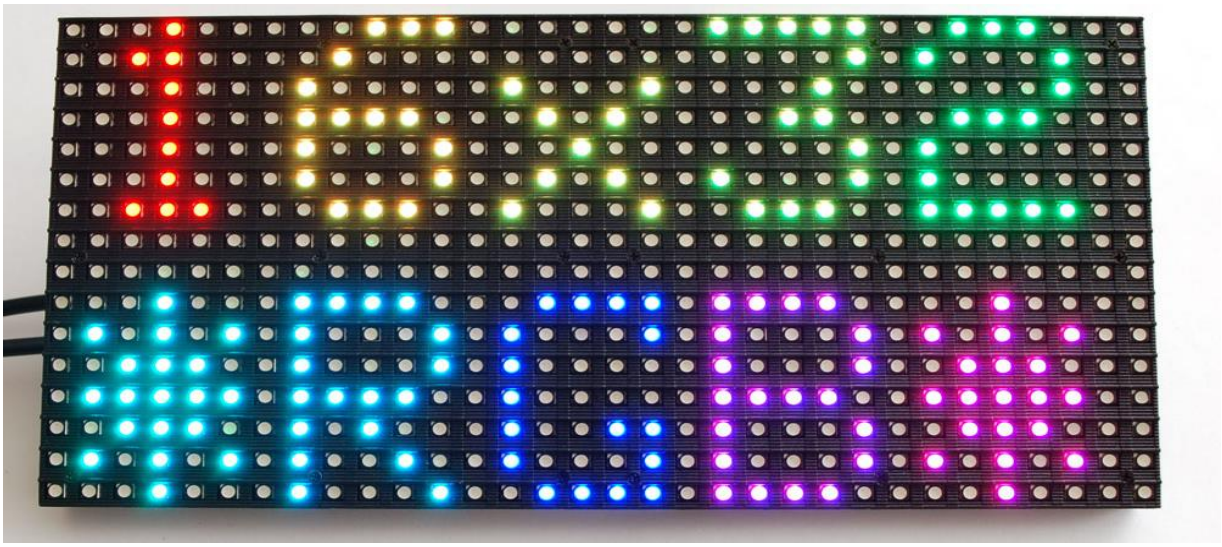

Рисунок. 2.29 – Світлодіодна матриця

Характеристики:

- Розміри: 192 мм x 96 мм x 12 мм
- Вага: 170 г
- 5V вхід з регулюємою потужністю, 2.5A макс
- 2000 MCD світлодіоди на 6 мм полі
- Критий дисплей, 150 градусів кут видимості

3. ХАРАКТЕРИСТИКА ПРОГРАМНИХ РІШЕНЬ ТА ІНСТРУМЕНТАРІЮ

3.1 Дослідження протоколів передачі даних у IoT

Взаємодія між пристроями всередині мережі суттєво залежить від протоколів для комунікації, які використовуються, розглянемо деякі з них.

3.1.1 Socket

Socket зазвичай використовуються для взаємодії клієнта з сервером. Типова конфігурація системи це коли сервер розміщується на одній машині, а клієнти - на інших машинах. Клієнти підключаються до сервера, обмінюються інформацією, а потім відключити.

Socket має типовий потік подій. У моделі орієнтований на встановлення з'єднань клієнт-сервер, Socket на серверній стороні чекає запитів від клієнта. Для цього, сервер спочатку налаштовую персональну адресу, яку клієнти можуть використовувати для пошуку сервера. Коли встановлено адресу, сервер чекає запитів від клієнтів. Обмін даними між клієнтом і сервером відбувається, коли клієнт підключається до сервера через Socket. Сервер виконує запит клієнта і відправляє відповідь назад клієнту. [14] Схема роботи наведено на рис 3.1.

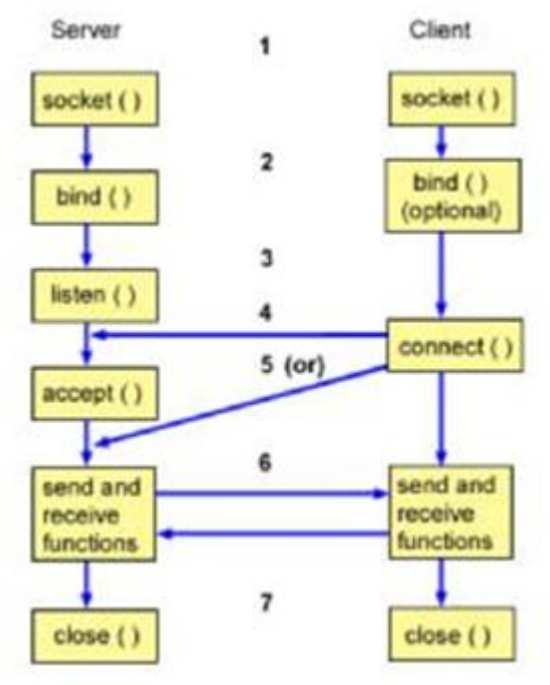

Рисунок 3.1 – Схема роботи Socket

3.1.2 REST

REST (передача репрезентативного стану) — це тип архітектури мережевих протоколів, що забезпечують доступ до інформаційних ресурсів. У 2000 році його описав та популяризовав один з засновників HTTP, Рой Філдінг. В основу REST закладено принципи функціонування Всесвітньої павутини і, зокрема, можливості HTTP протоколу.[15]

У протоколів REST дані мають передаватися у вигляді невеликої кількості стандартних форматів, таких як HTML, XML, JSON. REST протокол (як і HTTP) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформацію про стан між парами «запит-відповідь». Стверджується, що такий підхід забезпечує масштабування системи і дозволяє їй еволюціонувати з новими вимогами [16]. Приклад роботи наведено на рис. 3.2.

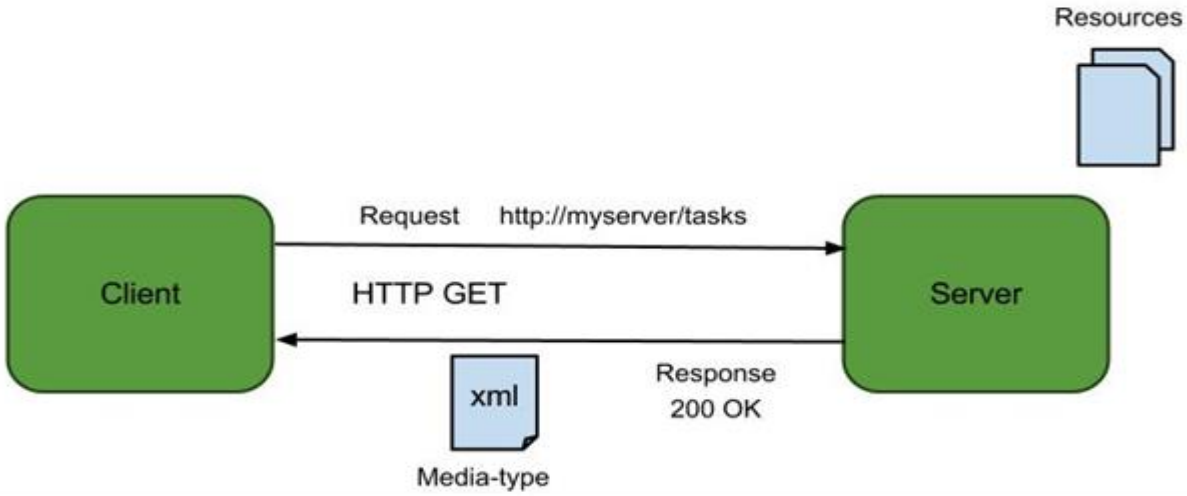

Рисунок 3.2 – Схема роботи REST

REST пропонує розробникам використовувати HTTP-методи явно відповідно до визначення протоколу. Цей основний принцип проектування REST встановлює однозначну відповідність між операціями створення, читання, оновлення та видалення (CRUD) і HTTP-методами. Згідно з цим відповідності:

• Для створення ресурсу на сервері використовується POST.

• Для вилучення ресурсу використовується GET.

• Для зміни стану ресурсу або його поновлення використовується PUT.

Для видалення ресурсу використовується DELETE.

3.1.3 MQTT

MQTT - це простий відкритий протокол, який був розроблений спеціально для застосування в IoT і обміну даними між пристроями. MQTT-мережа складається з MQTT-брокера, який виступає посередником у взаємодії MQTT-агентів - видавців і передплатників. Видавці публікують інформацію, призначену для передплатників. На рис. 3.3. наведено схему MQTT.

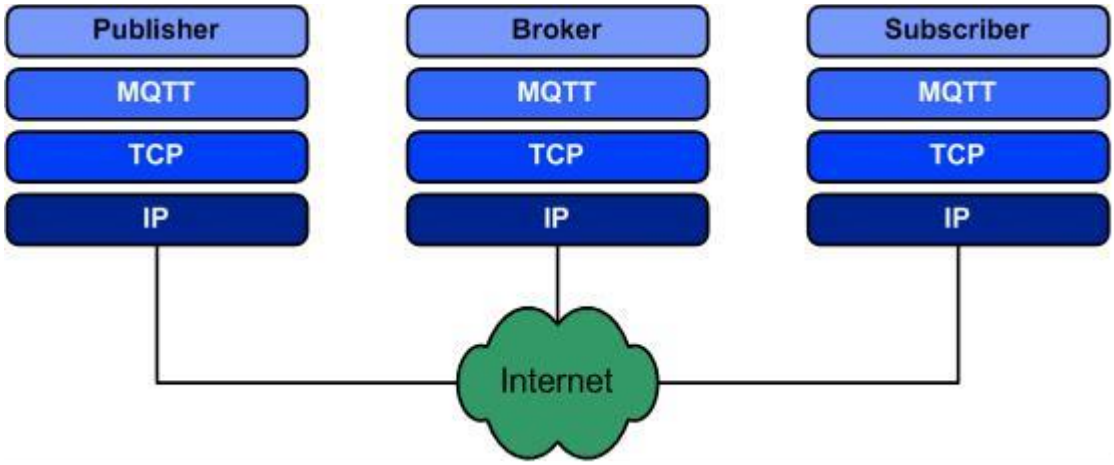

Рисунок 3.3 – Схема MQTT

MQTT працює за моделлю «видавець - передплатник», та використовує мінімальну кількість методів. Вони служать для вказівки дій, які необхідно виконувати. Ці дії зводяться до комунікації з брокером і до роботи з темами і повідомленнями. Агенти підключаються до брокера, а потім або публікують теми і повідомлення в них, або підписуються на теми і отримують повідомлення, в цих темах опубліковані. Завершивши роботу, агент відключається від брокера. Ось як виглядають методи MQTT:

- Connect - встановити з'єднання з брокером.

Відключити - розірвати з'єднання з брокером.

- Опублікувати - опублікувати тему на брокера.
- Підписка - підписатися на тему на брокера.
- Відмовитися від підписки - відписатися від теми на брокера.

Спрощена схема взаємодії між передплатником і видавцем з використанням MQTT-брокера наведена на рис. 3.4.

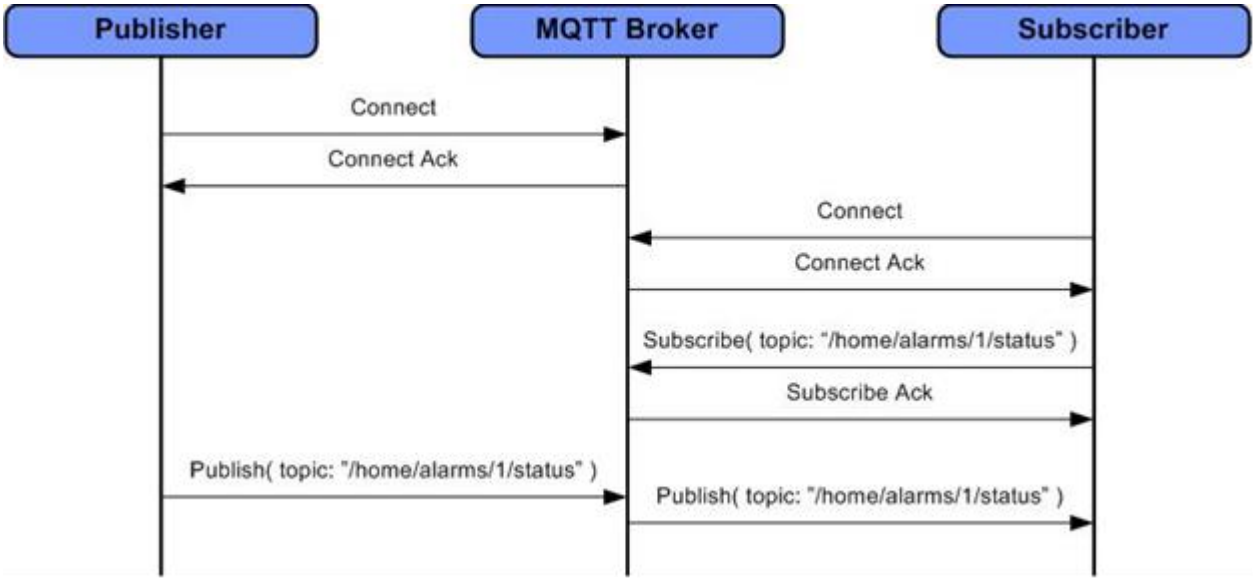

Рисунок 3.4 – Схема роботи MQTT

MQTT підтримує вказівку рівня якості (QoS Обслуговування). А саме, існують три таких рівні:

QoS 0. Цей рівень задіє стратегію «максимум одноразова доставка повідомлень». Приймач повідомлення не підтверджує їх отримання, відправник, відповідно, передає повідомлення лише раз, не роблячи спроб по їх повторної передачі. Це - метод «відправив і забув».

QoS 1. Тут застосовується підхід «мінімум одноразова доставка повідомлень". Гарантується, що приймач отримає повідомлення хоча б один раз. При цьому передплатник може отримати одне й те саме повідомлення кілька разів. А відправник буде робити повторні спроби відправки до тих пір, поки не отримає підтвердження в успішній доставку повідомлення.

QoS 2. Цьому рівню якості обслуговування відповідає найповільніша процедура доставки повідомлень, але при цьому він - найнадійніший. Його основна особливість - реалізація стратегії «одноразова доставка повідомлень» . При його використанні застосовується чотириступінчаста процедура підтвердження доставки повідомлень.

Вибір конкретного рівня якості обслуговування залежить від особливостей переданих даних і від того, наскільки важливо, щоб вони були доставлені. [17]

3.1.4 Modbus

Modbus - це послідовний протокол обміну даними, який з'явився в 1979 році і став стандартом де-факто для організації зв'язку між промисловими пристроями. MQTT з'явився на 20 років пізніше, але, не дивлячись на різницю у віці, спільне використання цих двох протоколів дозволяє дати вузькоспеціалізованим інтегрованим пристроям всі можливості, які доступні при підключенні до інтернету. На малюнку нижче показана схема взаємодії цих протоколів (Рис 3.5), а також - пристрій, який дозволяє таку взаємодію організувати - шлюз для інтернету речей.

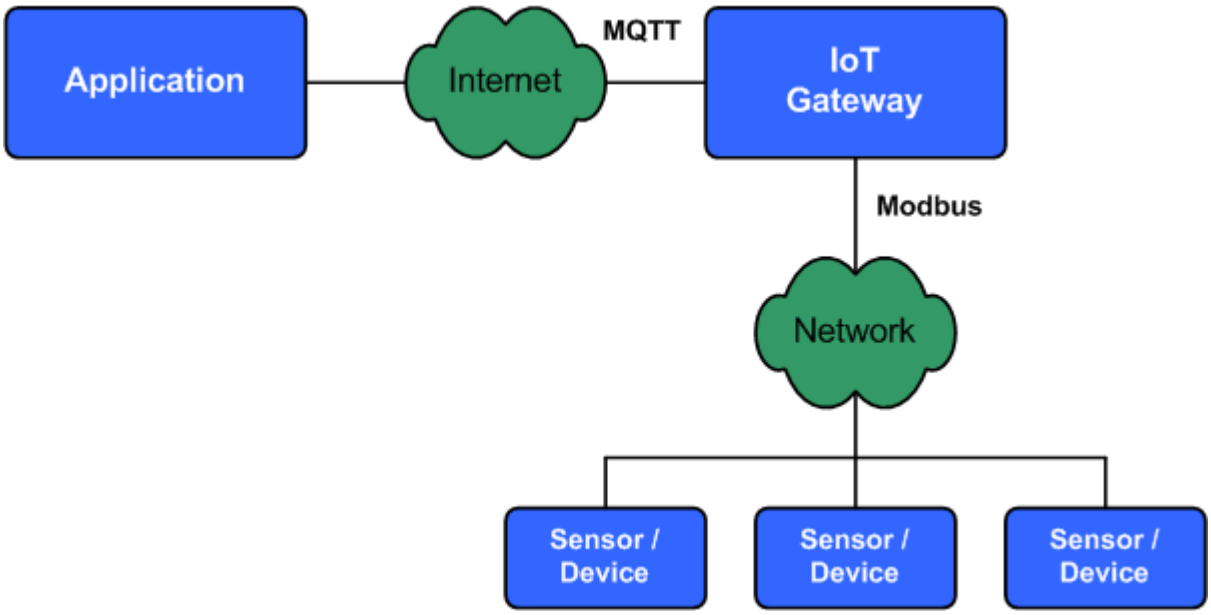

Рисунок 3.5 – Взаємодія протоколів MQTT та Modbus

Modbus, за десятиліття існування, розвинувся в широкий набір протоколів, що використовують різні способи фізичного зв'язку пристроїв (наприклад, інтерфейс RS-485). В основі всіх цих реалізацій - послідовний протокол, побудований за моделлю «ведучий - ведений». Провідний пристрій відправляє відомому запит, а ведене на цей запит відповідає. У стандартній мережі Modbus присутній один головний, провідний вузол і до 247 підлеглих -відомості. Однак, треба зазначити, що двухбайтовая система адресації здатна значно послабити це обмеження.

З використанням інтерфейсу RS-485 зв'язок між головними і підлеглими пристроями здійснюється за допомогою пакетів, які містять код функції і дані.

Цей код вказує на те, яку дію необхідно зробити, наприклад, на необхідність прочитати дискретні вхідні дані, прочитати дані з FIFO-черги, виконати діагностичну операцію. Підпорядкований пристрій, отримавши команду і виконавши її, відповідає, відправляючи пакет, структура якого дуже проста. Завдяки низькій обчислювального навантаження, яку створює взаємодія по Modbus, «спілкуватися» з його використанням можуть різні пристрої: від інтелектуальних контролерів до звичайних датчиків.

3.1.5 SNMP

SNMP - Simple Network Management Protocol, він же Простий Протокол Мережевого Управління. Протокол створений в 1988 році з метою управління великою кількістю мережевих пристроїв. З того моменту протокол набрав відповідну популярність і став стандартом. З моменту розробки протокол SNMP був 3 рази перероблений з версії SNMPv1, SNMPv2 і SNMPv3.

Крім управління пристроями, дуже часто SNMP використовують для моніторингу. SNMP може отримувати різну інформацію від будь-яких мережевих пристроїв, будь то роутер, свіч або просто комп'ютер в яких є підтримка даного протоколу

Мережа, що використовує SNMP для управління містить три основні компоненти:

- SNMP менеджер - ПО, яке встановлюється на ПК адміністратора (системи моніторингу)

SNMP агент - ПО, запущене на мережевому вузлі, за якими здійснюється моніторинг.

- SNMP MIB - MIB це Management information base. Цей компонент системи забезпечує структурування даних, якими обмінюються агенти і менеджери.

Схему взаємодії SNMP-агент - SNMP-менеджер наведено на рис. 3.6.[18]

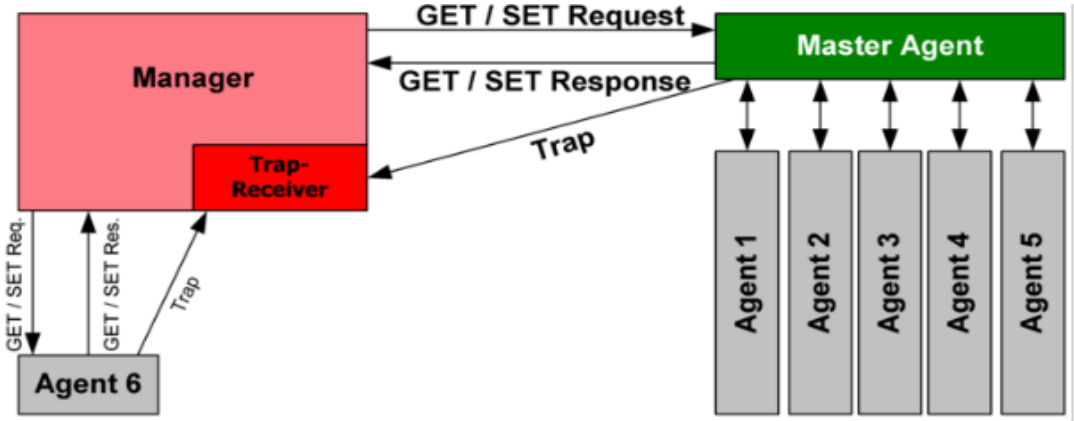

Рисунок 3.6 – Схема взаємодії SNMP-агент - SNMP-менеджер

3.2 Засоби та методи захисту інформації в IoT пристроях

Виконати захист інформації можна різними методами. Можна кодувати та декодувати повідомлення на пристроях IoT, але тут ми можемо зіткнутися з такою проблемою як недостатня швидкодія пристроїв, здебільшого мікроконтролерів. Інший варіант це покласти усі питання надійності та захисту передачі інформації на комп'ютерну мережу. В такому випадку ми вважаємо що мережа в якій знаходяться пристрої є досить надійною та ізольованою. Для підключення до віддалених частин мережі можливо використовувати VPN або SSH тунелі.

3.2.1 Захист Wi-Fi

Wi-Fi це один з найпоширеніших видів бездротового підключення до локальної мережі. Сьогодні він використовується майже усюди, тому його використання є найпростішим варіантом побудови IoT системи. В загалі, Wi-Fi є досить безпечний, якщо вимкнути усі проблемні його режими. Такими режимами є WEP шифрування там QSS. Обидва ці технології дозволяють без проблем отримати доступ до мережі і бажано їх не використовувати.

Найпоширенішим варіантом шифрування на даний момент є WPA2-PSK, який використовує AES шифрування. Злом точок доступу розглянуто у статті[19]. Автор вказує, що для злому можна використовувати лише повний перебір можливих паролів і такого перебору словника на 250 мільйони паролів, розміром 2 Гб, на ноутбуку, буде необхідно витратити близько 66 годин. У той же час,

словник на повних комбінацій 9 знакового паролю буде містити 900 мільйонів позицій і для його повного перебору необхідно декілька тижнів.

3.2.2 Обмеження доступу у протоколах передачі

Більшість протоколів для передачі даних підтримують аутентифкацію. Дуже часто для спрощення роботи її не використовують, але даремно, адже кожен з елементів безпеки може зробити більш складним життя хакера.

Отже, протоколи MQTT, REST, SNMP підтримують аутентифікацію, що також можна використати для забезпечення захисту інформації, що передається, та керування пристроями.

3.2.3 Модель підключення через VPN/SSH

VPN та SSH – це найпоширеніші засоби побудови тунелів.

У випадку використання VPN необхідно налаштувати мережеве обладнання, щоб воно виконувало маршрутизацію відповідним чином, або ж це може виконувати сам міні ПК.

У випадку використання SSH тунелів дані передаються через цей мережевий протокол. Для його використання є необхідним наявність деякого пристрою на якому вони будуть запущені та вказано переадресацію з яких портів виконувати.

Такий чином, обидва варіанти чудово підходять при роботі з міні комп'ютерами на ОС Linux.

Для забезпечення безпеки SSH є безліч методів, варіант у якому обмежується кількість невдалих підключень наведено у статі [20]. Він дозволяє суттєво ускладнити життя хакерам, які намагаються виконати атаку перебором паролів.

Схему підключення віддалених об’єктів до центральної мережі наведено на рис. 3.7.

У схемі підключення на міні ПК налаштовується тунель через VPN або SSH до централізованої системи, а датчики на базі мікроконтролерів через нього

підключаються до централізованого сервера або напряму через MQTT, або з використанням MQTT брокера.

У випадку використання MQTT брокера він відповідає за отримання повідомлень у своїй мережі та їх ретрансляцію до централізованого вузла.

Таким чином, можна прийти до висновків, що доцільним є покладання питань безпеки на стандартні засоби протоколів взаємодії та мережевої передачі даних. Також слід керуватися правилом, що безпеки мало не буває і не використовувати прості та легкі паролі, а також включати режими аутентифікації всюди, де це можливо.

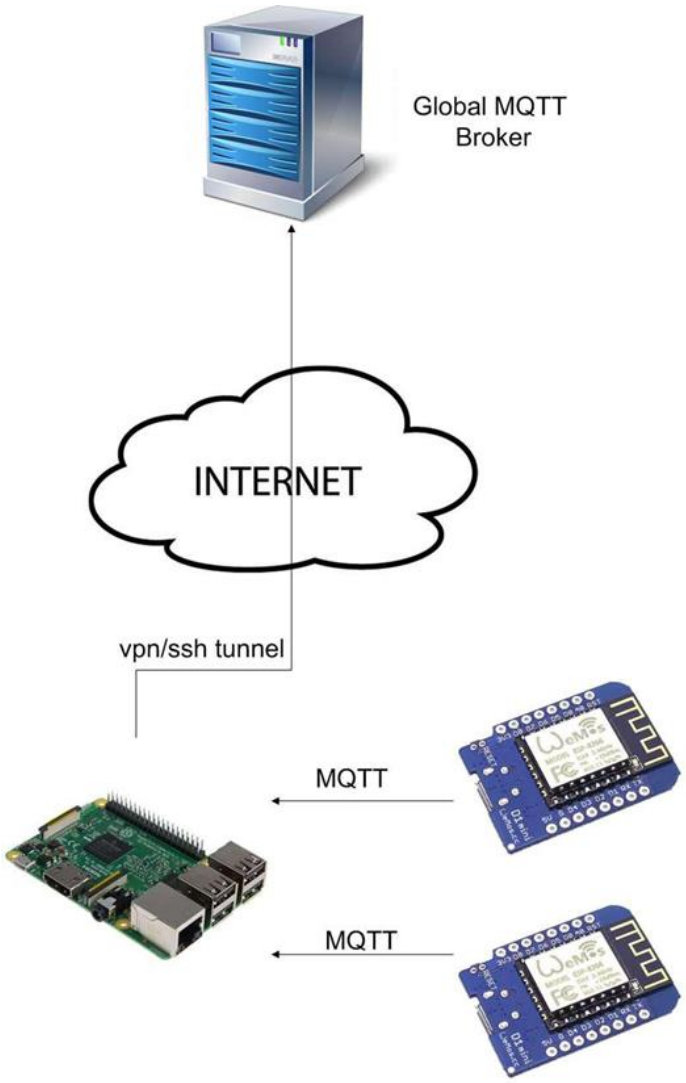

Рисунок 3.7 – Схема підключення віддалених об’єктів до центральної мережі

4. ОПИС РОЗРОБЛЕНИХ РІШЕНЬ ТА АЛГОРИТМІВ

4.1 Алгоритм роботи електронної черги

4.1.1 Зі сторони пацієнта

Відвідувач заходить в додаток видачі талонів і формує групу питань, натискаючи відповідні клавіші. Можливі клавіші підсвічуються іншим кольором. Повторне натискання на підсвічується клавішу скасовує вибір питання і клавіша гасне.

Питання сформовані в групи, які об'єднують кілька операцій під одним логічним назвою. Це зроблено для того, що б скоротити час вибору відвідувачем переліку питань. Перелік питань, а так же склад груп операцій, відповідних питань може бути змінений в будь-який час з робочого місця адміністратора системи.

Сформувавши список питань, відвідувач натискає кнопку «Отримати талон». Пульт вибору питань видає йому талон, на якому надрукований список питань, індивідуальний номер черги відвідувача і штрих-код, що відповідає цьому номеру Так само на талоні надруковані інструкції відвідувачеві, що він може зробити в період очікування черги, для прискорення процесу обслуговування (заповнити бланки, підготувати документи та ін.)

Відвідувач проходить в зону очікування і стежить за інформаційним табло. Система запрошує відвідувача повідомленням на табло:

Запрошений відвідувач підходить до робочого місця оператора і передає йому талон для реєстрації. Номер запрошеного користувача і номер підійшов відвідувача відображаються в робочому вікні програми. Якщо номери не збігаються, то оператор повинен відмовити громадянину у прийомі. Якщо номери співпадають, то в робочому вікні програми відображається інформація про котрий підійшов громадянина, з яких питань він підійшов і.т.д. Орієнтуючись на цю інформацію, оператор може правильно проінформувати відвідувача про його місці в черзі.

Якщо підійшов не той відвідувач або не підійшов ніхто, оператор натискає кнопку «Повтор».

Якщо до оператора не підійшов викликаний відвідувач і після повторного виклику, то оператор повинен натиснути клавішу «Відкласти». В цьому випадку система переносить номер викликаного відвідувача в відкладену чергу. Після цього система вичікує певний час (наприклад, 10 хв.) І повторює виклик відвідувача з відкладеної черги.

Якщо до оператора і після цього повідомлення не підійшов викликаний відвідувач, оператор знову натискає кнопку «Відкласти». Після цього система видаляє номер відвідувача з черги.

Таким чином, у відвідувача є три шансу потрапити на прийом до оператора: почувши перше запрошення, почувши повторне запрошення, після того як оператор натисне клавішу «Повтор», почувши третього запрошення, яке система зробить через 10 хв. Якщо все-таки відвідувач з тих чи інших причин пропустив свою чергу, то він повинен взяти новий талон.

У разі, якщо номер запрошеного і номер підійшов відвідувача збігся, то система виводить на сенсорну панель список операцій, за якими може бути обслужений відвідувач. Завершити обслуговування оператор може натисканням клавіші «Наступний». При цьому на екрані з'являється основна панель роботи оператора (оператора).

Схема взаємодії співробітників з клієнтами за допомогою електронної черги дає цілий ряд переваг:

- комфортне обслуговування для клієнтів: отримання необхідних довідок про послуги, що надаються, запис на прийом до співробітника під час, яке буде заздалегідь відомо і чітко визначено, можливість оцінки якості обслуговування;

- комфортна робота персоналу: завдяки статистичними даними, що надходять на адміністративний пульт, про кількість і щільності відвідувачів, керівництво має можливість розподілити робочий час своїх співробітників максимально ефективно і зменшити стресові навантаження.

4.1.2 Зі сторони адміністратора

У адміністратора є можливість в реальному режимі часу відслідковувати як загальне, так і деталізований стан черги - по операторам, по послугам, по часу обслуговування, за часом очікування в черзі і багатьма іншими параметрами. Це дозволяє приймати оперативні рішення щодо зміни необхідної і достатньої в даний час дня (або іншого періоду) кількості співробітників. Є можливість динамічно змінювати пріоритети обслуговування окремих талонів (наприклад, вставити певного клієнта «позачергово»), а також пріоритети обслуговування певної послуги певною точкою обслуговування. Наприклад, якщо є кілька точок обслуговування з «унікальною» послугою (або послугами), які можуть також надавати «звичайні» послуги, можна забезпечити пріоритетне обслуговування «унікальних» послуг такими точками.

4.1.3 Зі сторони керуючого персоналу

На підставі докладної статистики, яка накопичується системою управління електронною чергою, керуючий персонал може приймати обґрунтовані рішення по кадрам, кількості точок обслуговування, і т.п. Крім статистики, що має безпосередньо відношення до обслуговування черги, паралельно проводиться накопичення статистики по робочому часу кожного оператора (обіди, технічні перерви і т.п.) Система управління електронною чергою генерує безліч звітів на підставі яких можна приймати ті чи інші управлінські рішення.

Конструктор звітів. Ви можете створювати будь-які звіти, ґрунтуючись на статистичних даних, які автоматично надходять в систему.

Система сама контролює кінець робочого дня і, в залежності від налаштувань, контролює процес заняття черги останнім відвідувачем, ґрунтуючись на загальностатистичних даних, або переносить черговість на наступний день.

4.2 Підключення та ініціалізація світлодіодної матриці

Для підключення світлодіодної матриці до землі системи, необхідно з’єднати три контакти GND до відповідних контактів землі на Arduino. Підключення зображене на рис. 4.1

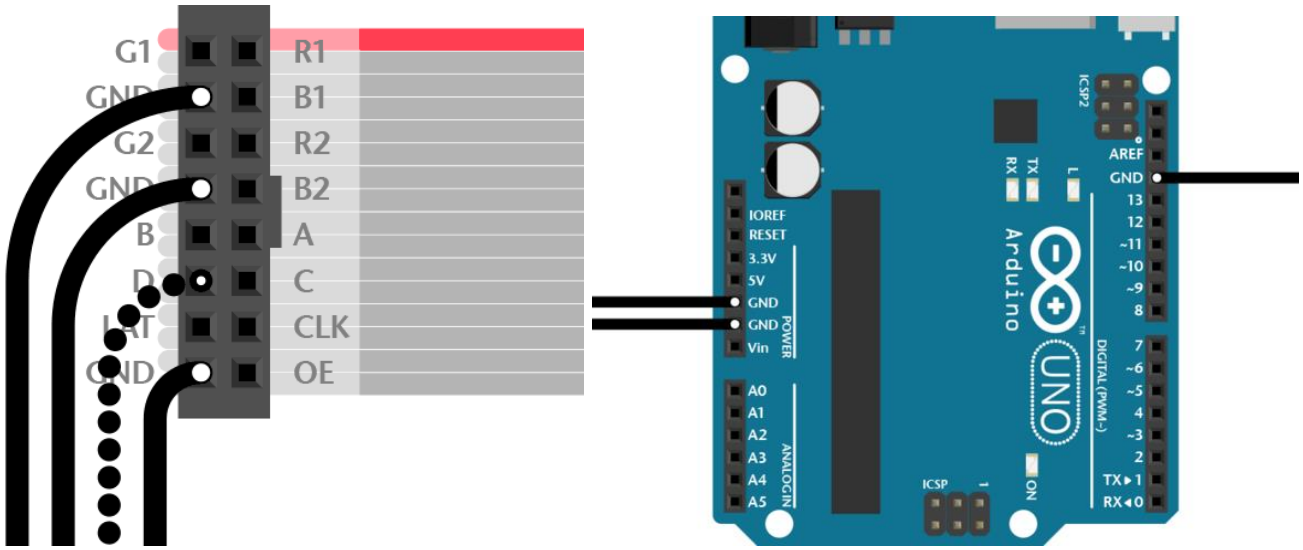

Рисунок 4.1 – Підключення GND контактів

Для підключення верхньої групи RGB контактів до Arduino, необхідно з’єднати контакти R1 з Pin2, G1 з Pin3 та B1 з Pin4. Підключення зображене на рис. 4.2

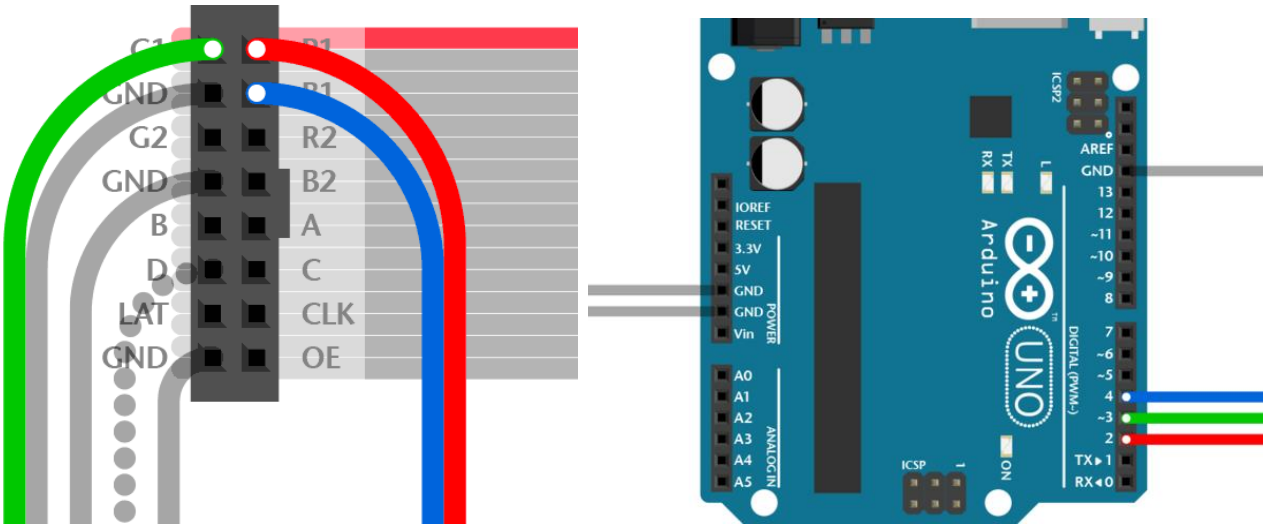

Рисунок 4.2 – Підключення верхньої групи RGB контактів

Для підключення нижньої групи RGB контактів до Arduino, необхідно з’єднати контакти R2 з Pin5, G2 з Pin6 та B2 з Pin7. Підключення зображене на рис. 4.3

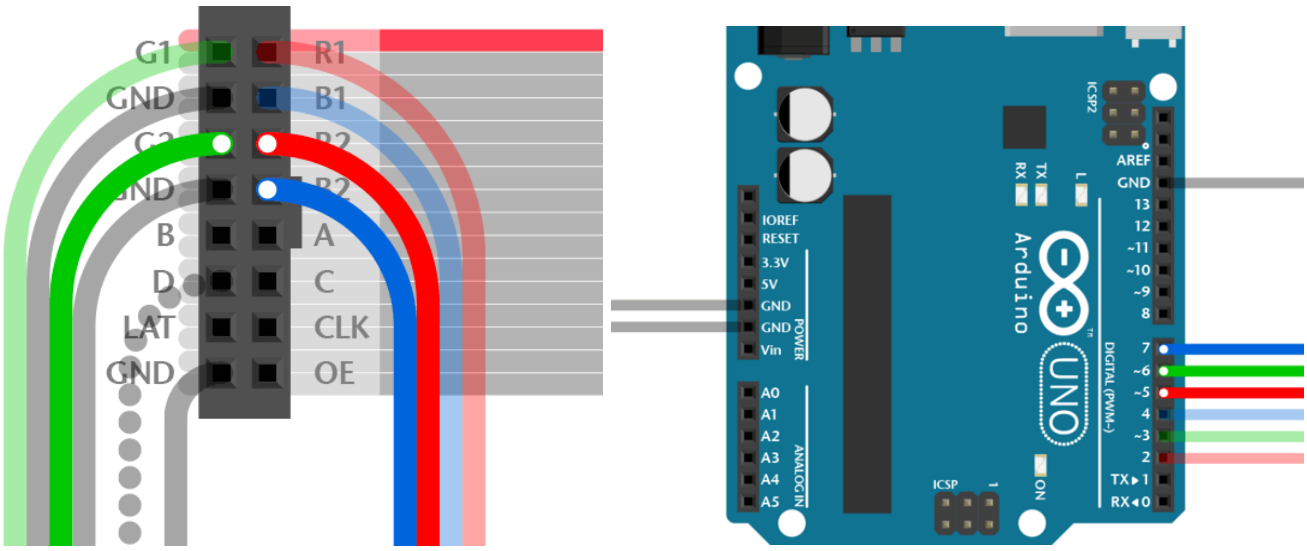

Рисунок 4.3 – Підключення нижньої групи RGB контактів

Для підключення контактів селектора ряду до Arduino, необхідно з’єднати контакти A з A0, B з A1 та C з A2. Підключення зображене на рис. 4.4

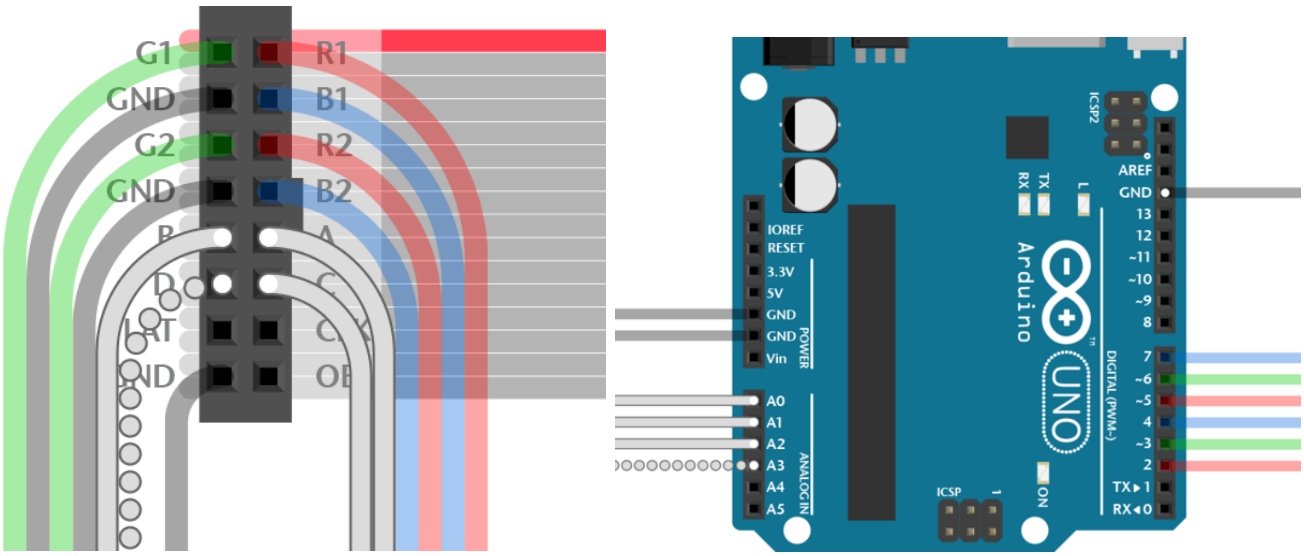

Рисунок 4.4 – Підключення селектора ряду

Для підключення службових контактів до Arduino, необхідно з’єднати контакти OE з Pin9, LAT з A3, CLK з Pin8. Підключення зображене на рис. 4.5

LAT інформує про кінець ряду інформації

OE надає дозвіл на вимикання діодів на пустих рядках

CLK надсилається с кожним бітом даних, синхронізація

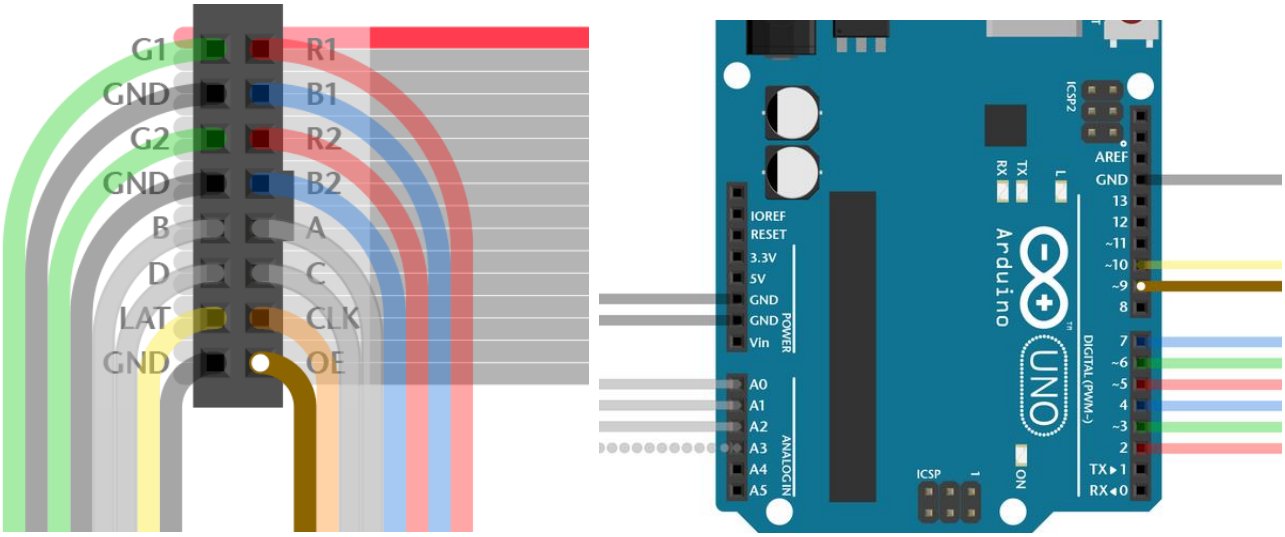

Рисунок 4.5 – Підключення службових контактів

Отже, світлодіодна матриця підключається до Arduino згідно таблиці 4.1

Таблиця 4.1 – Підключення світлодіодної матриці

Матриця	Arduino
GND	GND
GND	GND
GND	GND
R1	Pin2
G1	Pin3
B1	Pin4
R2	Pin5
G2	Pin6
B2	Pin7
A	A0
B	A1
C	A2
CLK	Pin8
OE	Pin9
LAT	Pin10

Підключення світлодіодної матриці до мікроконтролеру ESP8266 відбувається згідно рисунку 4.6

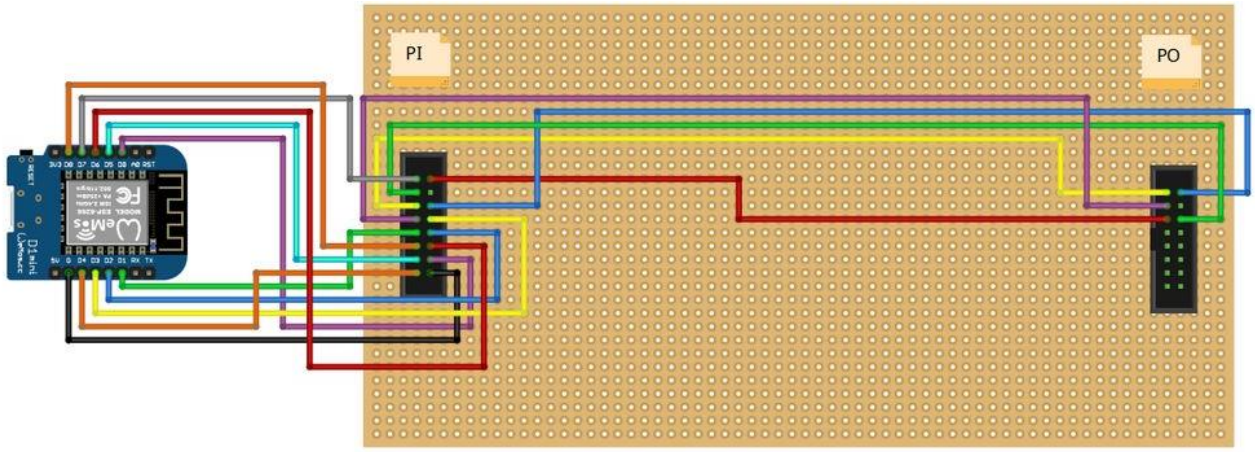

Рисунок 4.6 – Підключення матриці до ESP8266

Для перевірки роботи світлодіодної матриці, потрібно виконати просту керуючу програму на C++. В результаті її виконання, на табло висвітиться напис TEXT

```
#include <RGBmatrixPanel.h>
RGBmatrixPanel matrix(A, B, C, CLK, LAT, OE, true);
#define F2(progmem_ptr) (const __FlashStringHelper *)progmem_ptr

const char str[] PROGMEM = "TEST";
int16_t    textX         = matrix.width(),
           textMin       = sizeof(str) * -12,

void setup() {
  matrix.begin();
  matrix.setTextWrap(false);
  matrix.setTextSize (2);
  matrix.fillScreen(0);
  matrix.setTextColor (matrix.ColorHSV(0, 255, 255, true));
  matrix.setCursor(textX, 1);
  matrix.print(F2(str));
}
```

4.3 Підключення та ініціалізація годинника реального часу

Модуль годинника реального часу DS3231SN до мікроконтролеру відбувається згідно рисунку 4.7 та таблиці 4.2

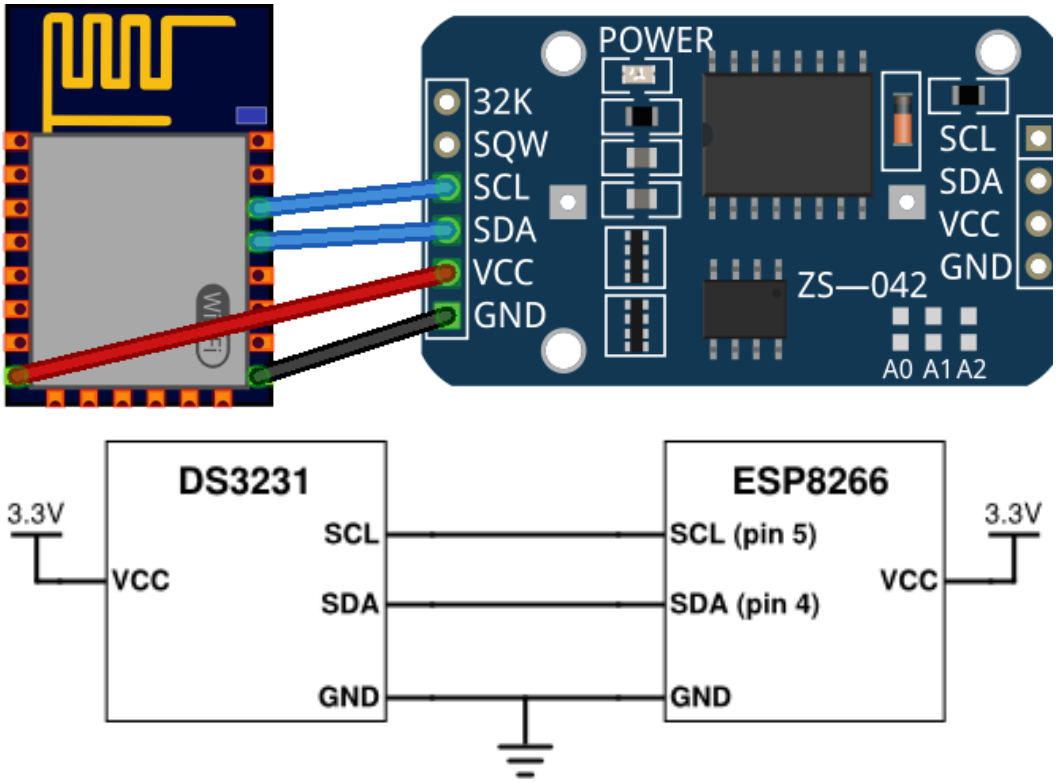

Рисунок 4.7 – Підключення годинника до ESP8266

Таблиця 4.2 – Підключення годинника до ESP8266

DS3231SN	ESP8266
SCL	Pin5 (SCL)
SDA	Pin4 (SDA)
GND	GND
VCC	VCC (3.3V)

Для налаштування поточного часу виконується наступна функція (повний текст керуючої програми для роботи з модулем годинника реального часу наведено у додатку):

```
void setup () {
    Serial.begin(115200); //Starts serial connection
    rtcObject.Begin();    //Starts I2C
    RtcDateTime currentTime = RtcDateTime(29,11,18,21,20,0); //define date
and time object
    rtcObject.SetDateTime(currentTime);                  //configure the RTC
with object}
```

4.4 Реалізація стандарту Modbus на Arduino

Для підключення Modbus пристрою до хабу, необхідно використати програмний модуль Modbus Tcp Binding. Цей модуль працює в режимі ведучого і забезпечує підключення декількох ведених пристроїв через послідовний порт або TCP/IP мережу. Для того щоб зв'язати з ним Arduino необхідно реалізувати в контролері ведений Modbus пристрій, скористаємося для цього бібліотекою Modbus-Master-Slave-for-Arduino.

Далі будуть розглянуть основні кроки необхідні для роботи з цією бібліотекою.

Всі функції бібліотеки реалізовані в одному файлі ModbusRtu.h. Для взаємодії з нею, в програмі потрібно створити об'єкт, задавши в його конструкторі Modbus адресу, номер послідовного порту, номер виходу.

```
Modbus slave(ID, 0, 0);
```

Далі необхідно визначити масив регістрів Modbus.

```
uint16_t au16data[11];
```

Після цього, при старті програми потрібно налаштувати послідовний порт веденого

```
slave.begin(9600);
```

В основному циклі програми необхідно викликати функцію обробки Modbus-повідомлень

```
state = slave.poll(au16data, 11);
```

Повний текст керуючої програми для налаштування протоколу Modbus наведено у додатку.

Для перевірки правильності налаштування мікроконтролера у формат slave, запустимо тестову передачу повідомлення з комп’ютера через емулятор master-

приладу. На рисунку 4.8 зображено налаштування з’єднання з емулятора. Далі необхідно змінити значення в регістрах HLD0 ... HLD2 і CL0 ... CL2, при цьому повинно змінитися значення у відповідному регістрі IN0..IN2 і DT0..DT2, потім натиснемо на кнопку макета, при цьому повинно змінитися значення в поле BTN, після натискання на поле LED , повинен загорітися або згаснути світлодіод. Вікно зміни регістрів у емуляторі зображено на рисунку 4.9

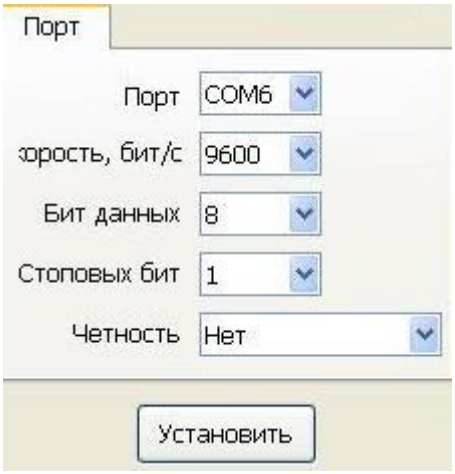

Рисунок 4.8 – Налаштування з’єднання з емулятором

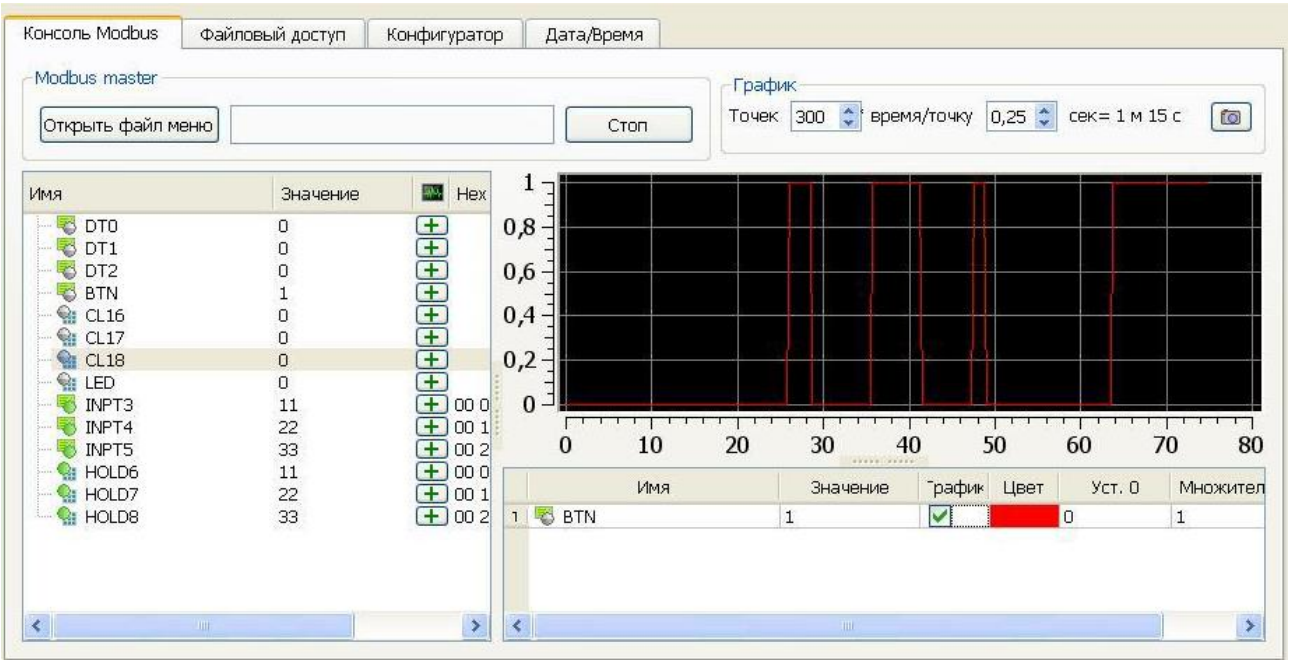

Рисунок 4.9 – Зміна та запис регістрів через емулятор

4.4 Реалізація протоколу MQTT на ESP8266

Для початку потрібно оголосити деякі глобальні змінні для з'єднання. Потрібні облікові дані Wi-Fi для підключення до мережі Wi-Fi.

```
const char* ssid = "YourNetworkName";
const char* password =  "YourNetworkPassword";
```

Також знадобиться інформація про сервер MQTT, а саме: адресу сервера, порт, ім'я користувача та пароль.

```
const char* mqttServer = "m11.cloudmqtt.com";
const int mqttPort = 12948;
const char* mqttUser = "YourMqttUser";
const char* mqttPassword = "YourMqttUserPassword";
```

Після цього необхідно оголосити об'єкт класу WiFiClient, який дозволяє встановити з'єднання з певною IP-адресою та портом. Також оголошено об'єкт класу PubSubClient, який приймає як вхід конструктора попередньо визначений WiFiClient.

Конструктор цього класу може отримати інше число аргументів, як це вказано в документації API.

```
WiFiClient espClient;
PubSubClient client(espClient);
```

Тепер, у функції налаштування, необхідно відкрити послідовний зв'язок, щоб вивести результат програми. Також на цьому етапі відбувається підключення до мережі Wi-Fi.

```
Serial.begin(115200);
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print("Connecting to WiFi..");
}

Serial.println("Connected to the WiFi network");
```

Далі потрібно вказати адресу та порт сервера MQTT. Для цього викликаємо метод setServer на об'єкті PubSubClient, передаючи як перший аргумент адресу та як другий порт. Ці змінні були визначені раніше, у константах.

client.setServer (mqttServer, mqttPort);

Потім використовується метод setCallback на тому ж об'єкті, щоб вказати функцію обробки, яка виконується при отриманні повідомлення MQTT.

client.setCallback(callback);

Тепер відбувається підключення до сервера MQTT, як і раніше, у функції налаштування. Спроби підключення до сервера в циклі, будуть відбуватись, поки не буде отримано успіх.

Для здійснення фактичного підключення використано метод підключення, який отримує як вхід унікальний ідентифікатор клієнта, який називатимемо "ESP8266Client", а також ім'я користувача та пароль для автентифікації, які були визначені раніше. Повернеться true при успішному з'єднанні і в іншому випадку false

У випадку невдачі можна викликати метод state на об'єкті PubSubClient, який повертає код з інформацією про те, чому з'єднання зникло.

```
while (!client.connected()) {
   Serial.println("Connecting to MQTT...");
   if (client.connect("ESP8266Client", mqttUser, mqttPassword )) {
    Serial.println("connected");
   } else {
    Serial.print ("failed with state ");
    Serial.print(client.state());
    delay(2000);
   }
}
```

Щоб завершити функцію налаштування, публікується повідомлення на тему. Для цього викликається метод публікації, який отримує в якості вхідних аргументів тему та повідомлення. У цьому випадку буде опубліковано повідомлення "Hello from ESP8266 " у темі "esp / test".

client.publish("esp/test", "Hello from ESP8266");

Потім ESP підписується на цю ж тему, щоб отримувати повідомлення від інших видавців. Для цього викликається метод підписки, передаючи як вхід назву теми.

client.subscribe("esp/test");

Після ініціалізації потрібно вказати функцію зворотного виклику, яка буде обробляти вхідні повідомлення для підписаних тем.

Аргументи цієї функції - це назва теми, корисне навантаження (в байтах) і довжина повідомлення.

У цій функції спочатку друкується назва теми на послідовному порту, а потім друкується кожен байт отриманого повідомлення. Оскільки також відома довжина повідомлення як аргумент функції, це можна реалізувати в циклі.

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived in topic: ");
  Serial.println(topic);
  Serial.print("Message:");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
  Serial.println("-----------------------");
}
```

Повна програма налаштування MQTT надана в додатку

Щоб перевірити налаштування MQTT, потрібно завантажте код із додатку та запустити його на ESP8266. Після запуску ESP8266 надсилає

повідомлення "Hello from ESP8266", яке не буде надруковане. Після цього ESP8266 підписує ту саму тему, на яку було надіслано повідомлення hello.

Якщо якийсь інший видавець надсилає повідомлення до теми "esp/test", то він буде надруковано, як показано на рисунку 4.10.

```
Connecting to WiFi..
Connecting to WiFi..
Connecting to WiFi..
Connecting to WiFi..
Connected to the WiFi network
Connecting to MQTT...
connected
Message arrived in topic: esp/test
Message:Hello from MQTTlens
-----------------------
Message arrived in topic: esp/test
Message:Hello again from MQTTlens
-----------------------
```

Рисунок 4.10 – Повідомлення у темі esp/test

Для тестування через MQTTlens перед підключенням ESP8266 підписався на тему "esp/test". Як видно на рисунку 4.11, було надруковано повідомлення "Hello from ESP8266". Після цього два повідомлення були надіслані до MQTTlens, що можна побачити на тому ж рисунку. Надіслані повідомлення - це ті, що показані на малюнку 2, які були отримані ESP8266, як очікувалося.

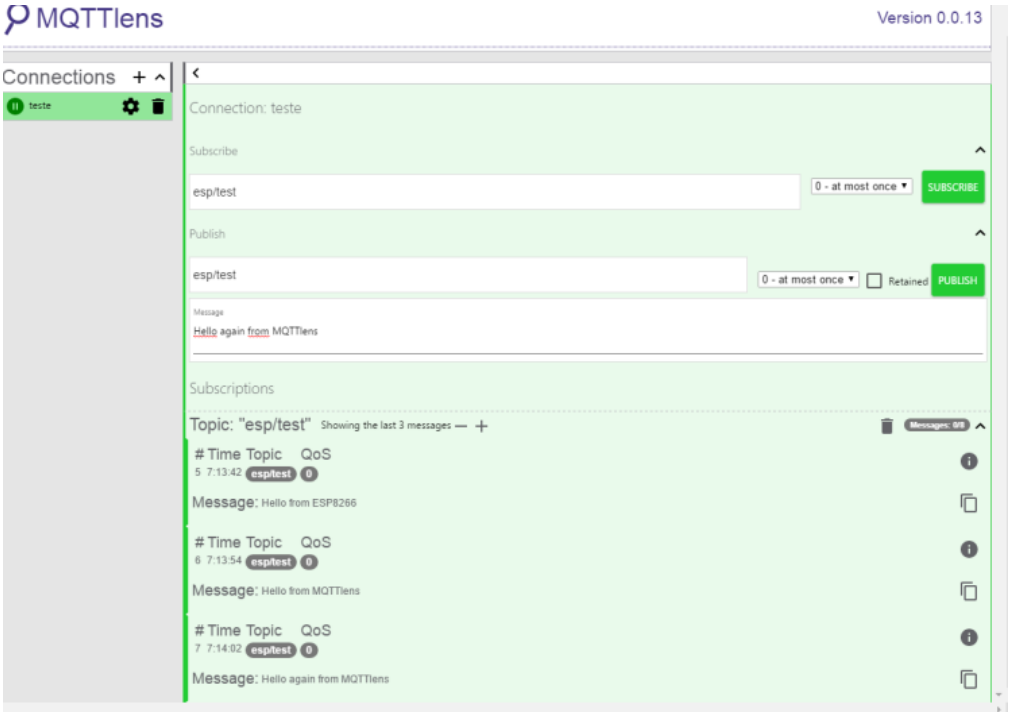

Рисунок 4.11 – Тестування програми у MQTTlens

5. ІНТЕРФЕЙСИ ТА АНАЛІЗ СИСТЕМИ

Користувацький інтерфейс розробленої системи складається з світлодіодних матриць (табло). На табло виводиться вся необхідна інформація про поточний стан черги. На рисунку 5.1 зображено інтерфейс виклику наступного пацієнта.

Рисунок 5.1 – Виклик наступного пацієнта

На інформаційне табло виводиться інформація про стан черги до певного кабінету (рис 5.2): хто наступний, хто після наступного. Це дозволяє пацієнтам орієнтуватись в часі очікування.

Рисунок 5.2 – Стан черги до кабінету

Також на інформаційне табло виводиться інформація про загальний стан черги у всій поліклініці. Який пацієнт стоїть у черзі до якого кабінету (рис 5.3)

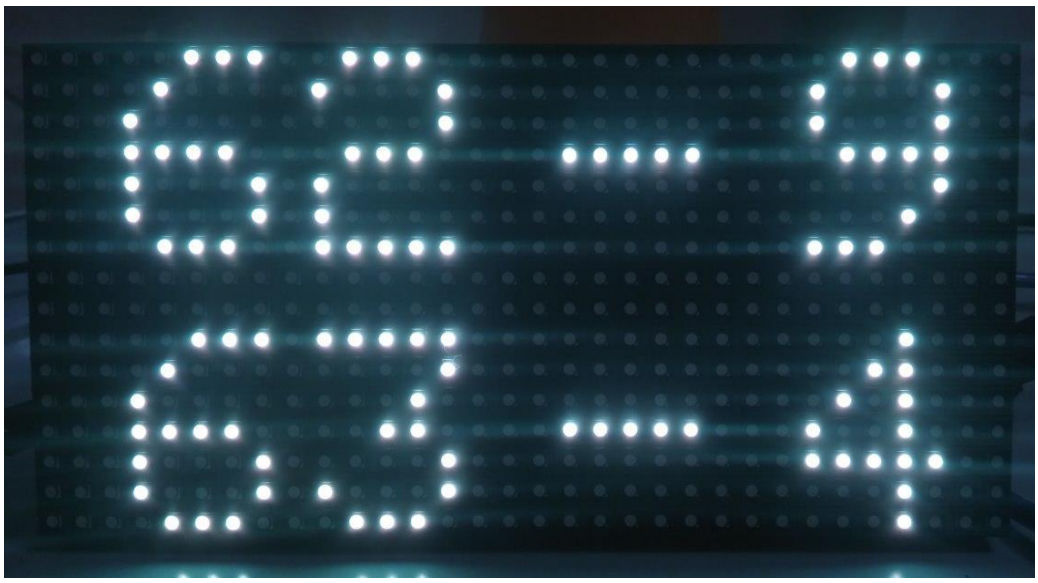

Рисунок 5.3 – Загальний стан черги

ВИСНОВКИ

В даній роботі було поставлено задачу провести аналіз можливостей різних моделей мікроконтролерів та міні комп'ютерів з метою їх використання для побудови веб-орієнтованої медичної системи.

Для реалізації поставленої задачі було обрано мікроконтролери від Arduino та ESP, а також міні комп'ютери Raspberry та Orange Pi. Порівняння обчислювальних можливостей виконувалося на мовах програмування: C++ для мікроконтролерів та міні комп'ютерів, Java та Python для міні ПК. Для порівняння було виконано 2 навантажувальні тести для кожної з мов програмування.

Перший тест полягав у складанні елементів масиву різного розміру, що дозволив оцінити швидкодію пристроїв. За результатами цього тестування можна прийти до висновків, що серед мікроконтролерів найкращий результат має ESP32, далі йде ESP8266. Серед міні ПК Raspberry Pi 3B показав найкращі результати у всіх 3 мовах програмування, результати Raspberry Pi B, який має ті ж самі характеристики, що й Raspberry Pi Zero є гіршими ніж Orange Pi Zero, який, в свою чергу, трохи відстає від Raspberry Pi 3B.

Другий тест полягав у виконані get запитів. Найкращий результат у мікроконтролерів у ESP8266, а серед міні ПК ситуація склалася аналогічним чином, що й з тестом 1.

Таким чином, можна прийти до висновків, що ESP8266 має достатню функціональність, потужність та досить не високу вартість, а враховуючи, вбудований модуль Wi-Fi, його можна вважати оптимальним рішенням серед мікроконтролерів для основи різноманітних датчиків.

Серед міні комп'ютерів ситуація склалась не так однозначно. Найкращі результати по більшості тестів має Raspberry Pi 3B, але його вартість значно вище рішень від Orange Pi та й до того присутні деякі проблеми з паралельністю на мові C++. Тому можна зробити висновок що найкращим варіантом для хаба пристроїв або міні сервера будуть рішення від Orange Pi з більш потужним апаратним забезпеченням, ніж модель Zero, їх вартість все одно буде нижче ніж у Raspberry.

Аналіз протоколів передачі даних показав, що найбільш доцільним при розробці системи є використання протоколу MQTT та Modbus. Так як MQTT вимагає менших обчислень на боці клієнта, тобто датчика, то його використання є більш бажаним. Що до безпеки передачі даних, було досліджено, що при належній організацій систем захисту (використання складних паролів) Wi-Fi підключення можна вважати безпечним, а побудову географічно віддалених систем доцільно виконувати з застосуванням технологій VPN та SSH.

Виходячи з цих результатів була спроектована та розроблена апаратна частина системи що дозволяє організувати роботу реєстратури та впровадити електронну чергу в медичних установах, зокрема у студентській поліклініці КПІ ім. Ігоря Сікорського. Для її реалізації було проведено аналіз роботи реєстратури поліклініки та аналіз переваг та недоліків існуючих систем медичної автоматизації. Розроблена система є простою для розуміння користувачем, зручною і не потребує значних грошових вкладень чи постійної технічної підтримки.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Качмар В. О. Медичні інформаційні системи – стан розвитку в Україні / В. О. Качмар // Український журнал телемедицини та медичної телематики. – 2010. – Т. 8., №1.- С.12-17.
2. Gurumurthi, S. & Benjaafar, S. (2004), Modelling and Analysis of Flexibile Queuing Systems, Wiley Periodicals, Inc. DOI 10.1002/nav.20020.
3. Чурпій І.К. Сучасний стан інформатизації в медицині / І.К. Чурпій, Н.В. Чурпій, В.Д. Скрипко // Буковинський медичний вісник. - 2011. - Т. 15, N 1. - С. 171-173.
4. Arduino Ethernet – Режим доступу: http://arduino.ua/ru/hardware/Ethernet
5. Arduino YUN – Режим доступу: http://arduino.ua/ru/hardware/YUN.
6. Arduino TITAN – Режим доступу: https://www.arduino.cc/en/Main/ ArduinoBoardTian.
7. D1 mini pro – Режим доступу: https://www.wemos.cc/product/d1-mini-pro.html.
8. SparkFun ESP32 Thing – Режим доступу: https://www.sparkfun.com/products/13907.
9. RASPBERRY PI ZERO – Режим доступу: https://www.raspberrypi.org/products/pi-zero/.
10. RASPBERRY PI 3 MODEL B – Режим доступу: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.
11. NanoPi - NEO – Режим доступу: http://www.nanopi.org/NanoPi-NEO_Feature.html
12. NanoPi-NEO-Air – Режим доступу: http://www.nanopi.org/NanoPi-NEO-Air_Feature.html.
13. Orange Pi Zero – Режим доступу: http://www.orangepi.org/orangepizero/.
14. IBM Knowledge Center - How sockets work – Режим доступу: http://www.ibm.com/support/knowledgecenter/ssw_ibm_i_71/rzab6/howdosockets.htm/.
15. REST – Режим доступу: https://uk.wikipedia.org/wiki/REST.

16. RESTful Web services: The basics – Режим доступу: http://www.ibm.com/developerworks/webservices/library/ws-restful/index.html?S_TACT=105AGX99&S_CMP/
17. MQTT и Modbus: сравнение протоколов, используемых в шлюзах для IoT Режим доступу : https://habrahabr.ru/company/intel/blog/304228/.
18. SNMP протокол (основы) – Режим доступу: http://www.kmax.name/linux/snmp-protocol/.
19. Взлом Wi-Fi WPA/WPA2– Режим доступу : http://variable.pp.ua/взлом-wi-fi-wpawpa2-для-чайников/
20. Защищаем SSH от брутфорса на любом порту – Режим доступу: https://habrahabr.ru/post/88461/
21. RESTful Web services: The basics – Режим доступу: http://www.ibm.com/developerworks/webservices/library/ws-restful/index.html?S_TACT=105AGX99&S_CMP/
22. MQTT и Modbus: сравнение протоколов, используемых в шлюзах для IoT Режим доступу : https://habrahabr.ru/company/intel/blog/304228/.
23. SNMP протокол (основы) – Режим доступу: http://www.kmax.name/linux/snmp-protocol/.
24. Взлом Wi-Fi WPA/WPA2– Режим доступу : http://variable.pp.ua/взлом-wi-fi-wpawpa2-для-чайников/
25. Защищаем SSH от брутфорса на любом порту – Режим доступу: https://habrahabr.ru/post/88461/